UNIVERSITY OF CALIFORNIA
SANTA CRUZ

OPTIMIZATION OF ITEM SELECTION WITH PREDICTION UNCERTAINTY

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in
Technology & Information Management

By

Liang Dai

March 2020

The dissertation is approved:

_____
Professor Ram Akella, Chair

_____
Professor Patrick Mantey

_____
Professor John Musacchio

_____
Dr. James Shanahan

_____
Quentin Williams

Acting Vice Provost and Dean of Graduate Studies

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Optimization of item selection with prediction uncertainty

by

Liang Dai

Selecting items from a candidate pool to maximize the total return is a classical problem, which is faced by people frequently in real life and also engineers in information technology industry, e.g., digital advertising, e-commerce, web search, etc. For example, web UI designers always try to find the best web design among many candidates to display to users, Google needs to select personalized engaging ads to display to users based on their historical online behaviors. Each of these industries has hundreds of billions of dollars market, which means that even a small performance improvement of item selection efficiency can drive hundreds of millions of dollars growth in the real world.

In these applications, the true value of each item is unknown and can only be estimated from observed historical data. There is a large volume of significant research about building prediction models which are trained on historical data to estimate the item values. Given data volume and computation resource restrictions, engineers choose different models, e.g., deep neutral network, gradient boosting tree, or logistic regression to solve the problems. We will not dive into this area too much in this dissertation. Instead, our focus is how to maximize the total return given these predictions, especially taking into account the prediction uncertainties for the value optimization.

In the large-scale real applications, the candidate pool can be extraordinary large. It is infeasible to pick some items from the pool to get interactive feedback for exploration. Actually, not only is exploration infeasible, but even estimating the value of each item

through a complex estimation mode is almost impossible due to the need of real-time response. For example, Apple needs to estimate users' favorite apps and recommends them to users when they visit Apple store. Google needs to select ads to display to users given a users' search queries. There are millions of candidates needing to be estimated from prediction models. It is very challenging to support such a large scale of model prediction under the low-latency constraint. Besides that, to have a good prediction accuracy, the models used in industry are getting more and more complex, e.g hidden neurons and layers of deep neural network increases rapidly in real applications, which also increases latency significantly. All of these make it infeasible to evaluate all candidates through one single complex model in large scale application. To solve this problem, engineers usually leverage the cascading waterfall filtering method to filter items sequentially, which means instead of using one complex model to estimate the values of all candidates, multiple stages are adopted to filter out candidates sequentially. For example, a simple model is used in the first stage to estimate candidates' values for choosing a small subset from all candidates. These selected items are then passed to another stage to be estimated by a more complex model. Intuitively, this cascading waterfall filtering method provides a good trade-off between infrastructure cost and prediction accuracy, which can save computational resources use substantially, and simultaneously select most promising items accurately. However, there is no systematic study about how to efficiently choose the number of waterfalls and how many filtered items in each waterfall. Engineers tune the settings of this system heuristically through personal experience or online experiments, which is very inefficient, especially when the system is dynamic and changes rapidly. In this dissertation, we propose a theoretical framework for the cascading waterfall filtering problem and

develop a mathematical algorithm to obtain the optimal solutions. Our method achieves a dramatic improvement in an important real-world application, which adopts cascading water filtering system to select a few items from tens of millions of candidates.

There are also some cases in which the candidate pool is relatively small. For instance, the number of web UI candidates is usually less than one hundred. Then, we are able to explore during item selection process. A typical exploration case is online experimentation, which is widely used to test and select items in real applications. In this situation, we can get interactive feedback to evaluate items. Considering online experiments for example, we usually randomly segments users into several groups, show them different candidates, and then compare the overall performance of each candidate to find the item with the largest value. Among all designs, A/B testing, which usually segments users into two statasitically equivalent groups to measure the difference between two versions of a single variable, is the most popular. For instance, in order to compare the impact of an ad versus another, we need to see the impact of exposing a user to viewing the first ad, and not the second, and then compare with the converse situation. However, a user cannot both see the first ad and not see it. Consequently, we need to create two "statistically equivalent populations" and expose users randomly to one or the other. This method is straightforward. However, the defect of this method is also obvious: to measure both versions, this method cannot expose all users to the best version, which leads to potential value loss. Some multi-armed bandit algorithms, e.g., Randomized Probability Matching (RPM), Upper Confidence Bounds (UCB), whose objective is maximizing the total return in experiment, have been proposed for improvement. However, these methods do not take into account the statistical confidence levels of the final result from the experiment and the

corresponding impact on the subsequent item selection in the post-experimental stage. To solve this problem, we develop algorithms to achieve a good trade-off between reducing statistical uncertainty and maximizing cumulative reward, which aims at maximizing the total expected reward of item selection over a total duration, which includes both the current experimental stage and the post-experimental stage. The proposed algorithms demonstrate consistent and statistically significant improvements across different settings, outperforming both A/B testing and multi-armed bandit algorithms significantly.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerest gratitude to my adviser, Ram Akella, for his great support of my Ph.D. study and research. I would like to thank him for his great encouragement when I started the research journey, for his tremendous guidance when I explored in research areas, for his huge inspiration to make me think critically and innovatively, and for him being a great friend in my life. The freedom I have had to lead and propose solutions in these collaborations is remarkable. These interactions have shaped my professional skills without a doubt.

I would like to thank my committee members, Professor Patrick Mantey, Professor John Musacchio, and Doctor James Shanahan. They gave me insightful comments and advices which make the dissertation more comprehensive and concrete. Besides, I would especially like to thank James Shanahan, who brought me into large scale machine learning industry, and provided me a rare opportunity to bring my machine learning knowledge to real-world products. I am deeply indebted to you.

I am so lucky that I joined a research group with lots of wonderful lab mates, Chunye Wang, Joel Barajas, Karla Caballero, Jing Du, Maria Daltayanni, and Shenshen Liang. Especially, Chunye and Joel helped me a lot in ram-up in my research. Their companionship and encouragement helped me through the many trials and tribulations of the tough doctoral journey. I am grateful for all my friends in UCSC.

I want to recognize and thank Emily Gregg, Lisa Stipanovich, Carol Mullane, Tracie Tucker, Ally Modzeleski, Arielle M.S. Freitas, Linda Harris, and Theo-Alyce Gordon for

helping me above and beyond the call of duty. My graduate school experience would not have been possible without their generous support.

Finally, I owe a great deal to my family. My parents Jianwu and Wanqin gave me a life, made tremendous efforts to bring me up, influenced me deeply with their great personalities. I want to thank my wife Shenshen, who always supported me to pursue my dreams unconditionally. Without her support, understanding and patience, I cannot make my Ph.D journey. This dissertation is dedicated to them.

CHAPTER I: Introduction

1.1 Item selection

How to make choices among many options from a candidate pool is a problem which we confront throughout our lives. When we were a baby, we selected toys which can please ourselves most from a basket. When we were a child, we chose the fruit which we liked most in a plate. When we grow up, we need to choose university, major, job, etc. When we approach the end your life journey, we still need to think about choosing someone to inherit our assets. The problem of choosing from a candidate pool accompanies us all along our entire lives.

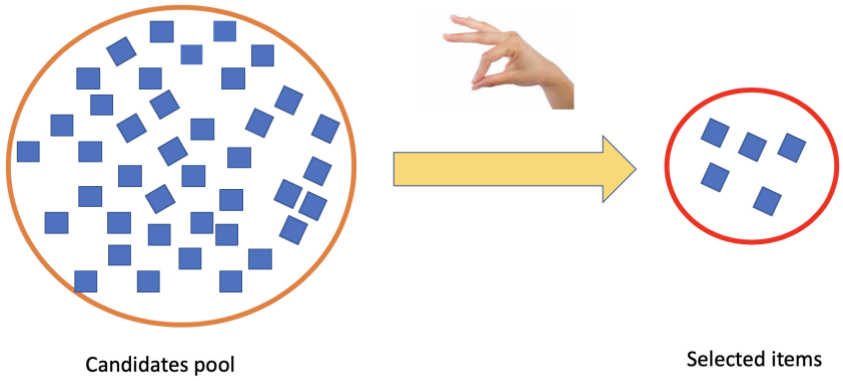


Figure 1. Item selection problem

From an economics perspective, the optimal choices depend on two factors: the reward associated with each candidate option, and the estimation of that reward. Different people have different reward functions to value each candidate. Some children prefer apples to oranges because apples taste better to them. In contrast, some others like oranges more due to their different preference, i.e., reward function. A personalized reward function is developed during an individual's growth, which is heavily influenced by personal experience and environment.

In some cases, we know our reward for each candidate well. Let us take the fruit comparisons for example. People have the chance to taste both apple and orange many times. Each fruit may taste a bit different, juicy or not, sweet or tart etc. So, they know which one is their favorite. Then the selection is very simple and straightforward. However, in some other cases, people do not have a clear estimation for each candidate. This is also very common in our life. People might hesitate in choosing to study math or computer science, starting working for a hedge fund or an IT company. The reason is that they do not know the "value" of each candidate very well. There are also two different scenarios. One is that you do not have any information about these candidates. The other one is that you have some information about them, which can help you improve the chance of making the right decision. In the first scenario, your decision would purely rely on blind guesses. In the second scenario, your decision would be optimal, or near-optimal if the information is used properly. In some situations, you even have the chance to try out some candidates and obtain better estimates for them, before making a final decision. For example, some high schools provide shadow visits to help students find out whether the school life is enjoyable for them. Though there are only one or two days to try it in person, the shadow visits do provide a chance for better evaluation. But unfortunately, most companies do not provide shadow visits. If you want to explore different jobs, you must expend significant time and efforts at your own expense: try a job for a few months, change to another one if you really do not like it. So, in this situation, how to make a good decision without any exploration is extremely important.

1.2 Research problems

As we discussed above, if we look at the item selection problem from the knowledge of "reward function" perspective, there are several different cases, which relate to three kinds of problems:

- The first kind is that the reward value of each item is deterministic and known. Then, the optimal solution for item selection problem is straightforward: the items ranked with higher reward should be selected. So, we do not discuss the problem in the dissertation.

- The second kind of problem is that reward of each candidate is totally unknown. And meanwhile, we do not have any information to estimate the reward. Then, there is no difference between selection policies. We do not talk about this problem either.

- The third kind of problem is that the reward of each choice is partially known, i.e., we have some estimates of these rewards. We will focus on discussing and solving this kind problem in my dissertation. There are also two different kinds of situations: explorable item selection and inexplorable item selection. Explorable item selection means that we can take actions and explore to obtain more accurate estimates, inexplorable item selection is that we can only rely on the current available information to select items. We will discuss these two kinds of problems in the following chapters. Independent of which one of these two scenarios is encountered, the prediction uncertainty for individual reward is important in the overall reward optimization, which would be our focus.

Figure 2 shows these different kinds of problems. We will focus our research on the green hatched area in this dissertation.

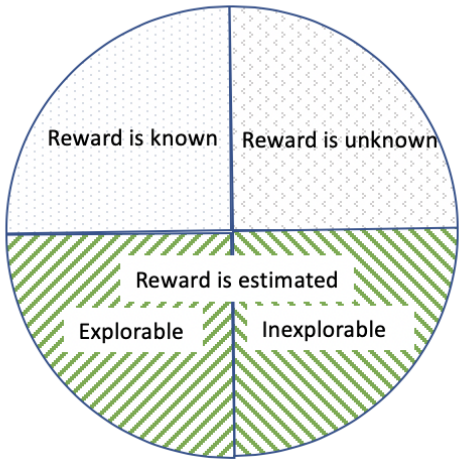


Figure 2. Different item selection problems (Green hatched area is our research focus.)

1.3 Contributions of the dissertation

In this dissertation, we would like to discuss item selection problem given the value estimation for each candidate, and especially focus on the discussion of two scenarios: explorable situation and inexplorable situation.

The inexplorable situation seems to be simpler as we only make the decision given the estimated value of each item. And that is true if we only select items through one stage. However, it is almost impossible to select items through one single ranking stage in real-world applications if the candidates pool is large. It is very challenging to support such a large-scale model prediction with low-latency response requirements. Besides that, to obtain a good prediction accuracy, the models used in industry get more and more complex. Deep neural networks with many layers and neurons are widely used these years. All of these make it infeasible to evaluate all candidates through one complex model in the large-

scale usage scenarios. To solve this problem, engineers leverage the cascading filtering method, which is shown in Figure 3. This cascading filtering method is widely used in real applications. However, as we discussed in the section above, it has not been well studied yet. Though there are some similar problems, such as how to efficiently and sequentially sample, rank, and select items among a fixed set, which are discussed in statistic community and simulation community, there is no deep analysis of the problem given only predicted values and the uncertainties of the predictions. One of the reasons is that researchers usually assume improving model prediction accuracy is the only way that can improve the system performance, and they take it for granted that the output of predictive models should be ranking score of items. However, they ignore the bias incurred by the prediction uncertainty and the item selection process, which we will discuss more in the next section. We are also going to define this problem and discuss a process to develop a better scoring function for item selection, and to find the optimal settings for a cascade ranking and filtering system.

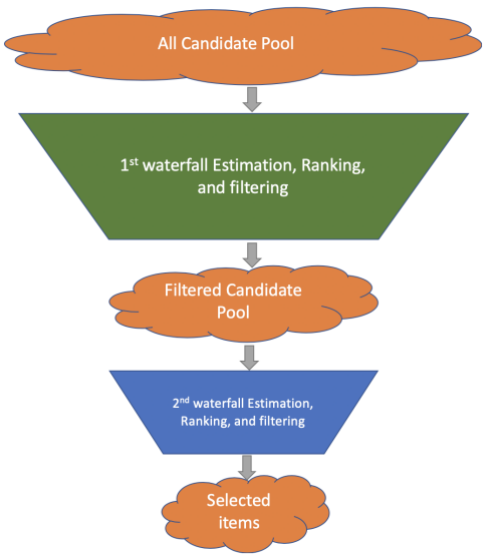


Figure 3. A typical example of a cascading ranking and filtering system

When the items are explorable, we need to tackle the problem of "exploration" vs. "exploitation" tradeoff, which is similar to the multi-armed bandit problem. However, there is no restriction on the degree of exploration in a typical multi-armed bandit problem, which is unfortunately not true for many real-world applications. In most cases, items are only explorable over a finite duration. After this exploration stage, we need to make the decision of selecting items. For example, you cannot always obtain an unbounded number of shadow days in schools. After a few days, you must make the decision to join one. Another example is UI design test. Different UI designs can be tested during a period on the website. But after the test period, they must eventually make the decision of whether the new design should be rolled out. Keeping two or more designs not only increases the cost of maintenance, but also incurs inconsistent experience among users. How to make a good trade-off during limited experiment time will be discussed in this dissertation.

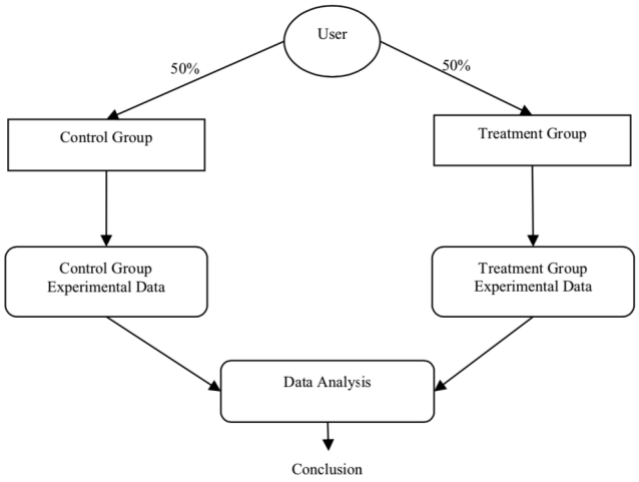


Figure 4. A typical A/B testing method for exploration

Compared with multiarmed bandit algorithms and typical online testing method, such as A/B testing, our proposed algorithms can handle "exploration" vs. "exploitation" trade-off

much more efficiently in terms of total reward. It can achieve a better trade-off between reducing statistical uncertainty and maximizing cumulative reward, which can significantly improves the total expected reward over a total duration, including both the current experimental stage and the post-experimental stage. We will demonstrate the improvement through our experimental results later.

Table 1: The comparison of A/B test, Multi-armed bandit algorithm, and proposed algorithms

	Maximize expected reward in experimental stage	**Maximize expected reward in post-experimental stage**	**Maximize expected reward in both experimental stage and post-experimental stage**
A/B testing	No	Yes	No
Multi-armed bandit algorithms	Yes	No	No
Proposed algorithms	NO	No	Yes

1.4 Outline

In the rest of this dissertation, we will first discuss inexplorable item selection, talk about how prediction uncertainty changes the optimal policy for a cascading selection system, and how to optimize the reward given the estimates of item values. Then, we will discuss explorable item selection, and develop the algorithms to achieve a better trade-off between exploration and exploitation for the online experiment of item selection. Finally, we will discuss our conclusions, and future work in this area.

CHAPTER II: Inexplorable item selection optimization

As we mentioned above, there are many cases in which we cannot explore items during the selection process. The infeasibility of exploration makes the item selection policy wholly depend on the estimation of the value for each item. The problem is very straightforward if we only have one single ranking and selecting stage. However, using one single stage to rank and select items is unrealistic in a large-scale application due to the infrastructure limitations. In that situation, engineers have to use multiple stages to rank and select items, which is called a cascading selection system.

Cascading selection systems are widely used in real applications to solve the problem of item selection from a huge candidate pool under resource. In this chapter, we discuss the cascading selection system used in industry and the associated optimization problem. We would like to find the answers of the following two questions: 1. How to find the optimal system settings, e.g. the number of stages and the number of items filtered at each stage? 2. How to select items given predictions at each stage? For the second problem, we mainly discuss the binary event reward use case, which is a more complex problem to solve in industry. After that, we will talk about how to take CPU and latency cost into consideration for the optimization.

2.1 Single stage selection

If we only use one single stage to rank and select ads (as shown in Figure 5), the optimal selection policy would be very straightforward: as we only care about total expected returns, we should always select items with larger expected or predicted values, which is not worth discussing from a research perspective.

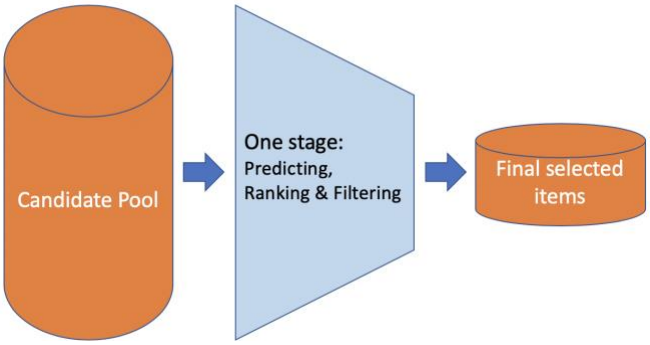


Figure 5. A single stage selection

2.2 Multiple stage selection

In real-world applications, there are many problems that concerns selecting a subset of items from a huge candidate pool. For example, Google needs to select ads to display to users after receiving a user's search query. Apple needs to estimate users' interested apps and recommends personalized apps to users when they visit the App Store. To solve this kind of problem, machine learning engineers usually build predictive models to estimate the value of each candidate, sort candidates by these predictions, and then select the top ranked candidates to be displayed. The size of candidate pool can be very large, especially for these large-scale applications. For example, there are around 2 million available apps in app store [1]. And in 2015, there have been already 4 million advertisers running ads on Google [2]. Also, many users visit Google website and App Store concurrently. It is very challenging to support a large-scale model predicting with low-latency response. Besides that, to achieve a good prediction accuracy, the models used in industry get more and more complex. For example, deep neural network models get very complex: many models used in real-world applications have a large number of layers and nodes. To have a high accuracy, the model size sometimes could be hundreds of Gigabytes or even larger. All of these make

it infeasible to evaluate all items through complex machine learning models in one single stage.

To solve this problem, engineers adopt a cascading selection method. Instead of using complex models in one single stage to estimate, sort and filter all candidates, multiple stages are used to filter items sequentially. Figure 6 is an example of a two-stage cascading filtering system. A simple model is used in the first stage to estimate candidates' values. And then we sort and filter candidates to get an intermediate subset. These intermediate candidates are then passed to the second stage to be further estimated by more complex and more accurate models. We leverage the second stage predictions to select the final selected items. Intuitively, this cascading filtering method usually can attain a good tradeoff between performance and infrastructure cost, which can both select most promising items accurately and save many computation resources. However, there is no systematical analysis on how to choose the number of stages and how many filtered items in each stage. Engineers usually tune the system settings through experience and experiment. We would like to formulate and discuss these problems in this chapter. As for the selection criteria, engineers and researchers usually believe the expected values of items should be the golden rule for ranking items. But is that true? Do the items with the same expected values but different prediction uncertainties provide us the same return? We will start from some simple examples, and then further dive into the mathematical analysis of this interesting problem. In short, our conclusion demonstrates the invalidity of the "golden rule" in most people's mind and leads to the following conclusion: the optimal selection policy should treat items with the same expected values but different prediction

uncertainties differently, which means we should take into account prediction uncertainty in the optimization of cascading item selection system.

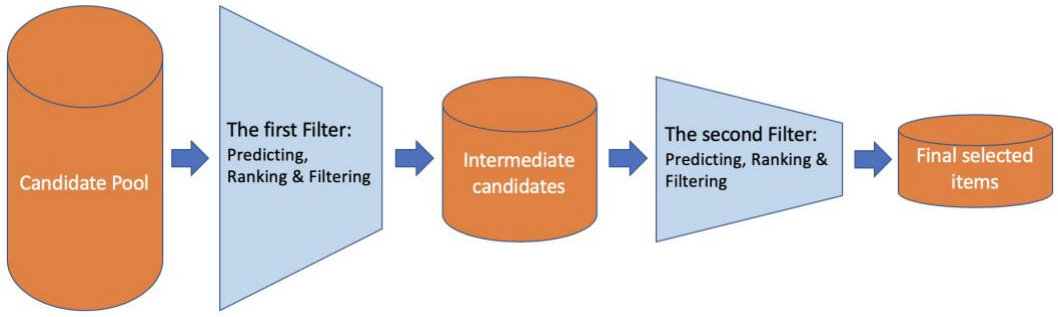


Figure 6. An example of multistage selection problem

2.2.1 Related work

This cascading filtering method is widely used in real-world applications. However, as we discussed in above section, it has not been well studied yet. There are some similar problems, such as how to efficiently sequentially sample, rank, and select items among a fixed set, which are discussed in statistic community [3] [4] [5] and simulation community [6] [7] [8]. For example, paper [4] discusses natural terminal decisions, i.e., decisions which are made in terms of the largest sufficient statistics in sequential selection procedures, to find good populations. Paper [5] talks about the situation that a sample of people independently examine a fixed set of k items and then rank these items according to personal judgment. The process of ranking the items is decomposed into $k-1$ stages. In the forward model, the most preferred item is selected at the first stage, the best of the remaining items is then selected at the second stage, and so on until the least preferred item is selected by default. In that problem, researchers care about the orders of selected items, which makes it closer to a ranking problem instead of a selection problem. However, none of them discusses the problem of total value optimization for the selected subset given the

predicted values without any further sampling opportunity. Our guess of why researchers have not paid much attention to the sequential item selection problem which we describe here is that previously there were few large-scale applications for a long time due to computation power constraint. Another possible reason is that engineers and researchers usually intuitively believe that improving model prediction accuracy is the only approach to improve the system performance and take it for granted that the output of predictive models should be the best score for item ranking. They ignore the survival bias incurred by prediction uncertainty and item selection process, which we will discuss in the next section. Firstly, we are also going to define cascade ranking and selecting problem, explore the optimal scoring function, and discuss how to find the optimal settings for system.

2.2.2 Problem definition

Let us first define a subset selection problem. Suppose we have *n* candidates with the true value of y_i, $i = 1, 2, \dots, n$. y_i is actually unobservable with i.i.d. Gaussian distributed $y_i \sim N(\mu, \sigma^2)$, where μ and σ^2 are known. We also briefly discuss some example of independent situations in Appendix C. The goal is to select a subset S_k which contains *k* items to maximize the total expected rewards:

$$V(S_k) = E\left(\max \sum_{i \in S_k} y_i\right). \tag{1}$$

And what we can observe are the predictions from ranking models $\hat{y}_{i,j}$, *j*=1, 2, ..., *T*, where *j* represents the prediction from j_{th} stage ranking model and *T* is the number of the stages. Here we assume that

$$\hat{y}_{i,j} = y_i + \varepsilon_{i,j}, \tag{2}$$

where $\varepsilon_{i,j}$ is the prediction error with Gaussian distribution $N(0, \sigma_j^2)$. And the noises are independent with each other within a stage. Here, the mean value off the error is 0, which means that the predictor at each stage is unbiased. We know that the total prediction error contains two components: bias and variance. Though sometimes regularization is introduced in model training to make a better trade-off between bias and variance to decrease the total prediction error, the bias is usually relatively much smaller compared with variance, especially for deep neural network, which usually adopts dropout for regularization. After looking at some real predictors used in some large real-world applications, we find that the bias component only accounts for less than 2% of the total prediction error. So, the unbiased assumption is still close to the real-world use cases.

The value of σ_j^2 shows the prediction uncertainty of the model at stage j. The larger σ_j^2 , the less prediction uncertainty. From equation (2), it is easy to get that $\hat{y}_{i,j} \sim N(\mu, \sigma^2 + \sigma_j^2)$. It is common to divide uncertainty into two types, aleatoric and epistemic uncertainty [47]. Aleatoric uncertainty refers to uncertainty about an inherently variable phenomenon. Epistemic uncertainty refers to uncertainty arising from lack of knowledge. The aleatoric uncertainties of the models at different stages are usually the same as long as all the models are trained on the same data set. However, more complex models can usually cover more knowledge, e.g. more informative features, more complex model structure, which lead to less epistemic uncertainties. As the model is more complex at later, the total prediction uncertainties of models at earlier stages are larger than the models at later stages $\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_T^2$. There are several methods to calculate the prediction uncertainties, e.g. Bayesian logistic regression, drop-out based uncertainty measure for deep neural network.

At each stage, only top k_j items are filtered and allowed to enter the next stage. The last stage chooses k items. So, we have $n > k_1 > k_2 > \cdots > k_T = k$. The optimization problem is how to find out the best system settings, i.e., the number of stages and the number of items selected at each stage.

$$V_{max}(S_k) = \max_{T,k_j,j=1,2,..T-1} V(S_k). \tag{3}$$

2.2.3 Prediction inconsistency caused by item selection

Before discussing how to find the optimization solution for problem (3), let us talk about the impact caused by the cascading selection process, which is usually ignored by most of researchers. With the assumption above, the prediction models at each stage are all unbiased. So, it is intuitive that the expected difference between predictions at two stages would be the same for a particular ad. But is it true? Let us run a simulation and see what happens. Figure 7 is the plot of the prediction distributions for a two-stage ranking problem. It is observed that the mean values of predictions from two stages are different, which is counterintuitive for most of researchers and engineers.

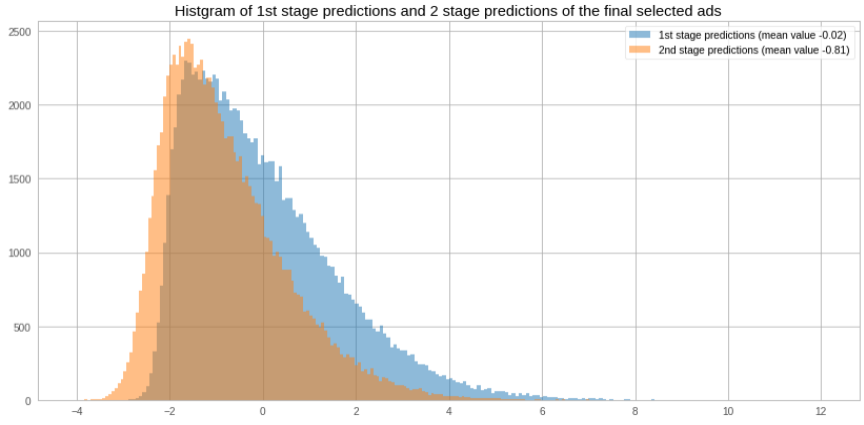


Figure 7. An example of the comparison of the predictions from two stages, $\mu = -8$, $\sigma^2 = 9$, $\sigma_1^2 = 4$, $\sigma_2^2 = 1$, $n = 1000$, $k_1 = 50$, $k_2 = k = 20$

To unveil this inconsistency and understand cascading selection process, let us start from a simple two-stage case. If the second stage is a dummy stage, which does not take any filtering effect. Then, the settings are fixed as $T = 2$ and $k_1 = k_2 = k$. Without loss of generality, let us assume the item index i is in descending order of the predicted value $\hat{y}_{i,1}$. Now, let us look at the distribution of the predictions at the first stage $\hat{y}_{i,1}$

$$P_{\hat{y}_{i,1}}(x|\mu, \sigma^2, \sigma_1^2, n) = \frac{n!}{(n-i)!(i-1)!} \Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)^{n-i} \cdot \left(1 - \Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)\right)^{i-1} \cdot \phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right), \tag{4}$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ and $\Phi(x) = \int_{-\infty}^{x} \phi(z)\, dz$.

When n is large enough, $\hat{y}_{i,1}$ is close to Gaussian distribution (see Appendix A for a detailed derivation)

$$N\left(\mu + \sqrt{\sigma^2 + \sigma_1^2} \cdot \Phi^{-1}\left(\frac{n-i+1}{n+1}\right), \frac{i \cdot (n-i+1)}{(n+1)^2 \cdot (n+2) \cdot \left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}\right). \tag{5}$$

So, the expectation of $\hat{y}_{i,1}$ is

$$E\left(\hat{y}_{i,1}|\mu, \sigma^2, \sigma_1^2, n\right) = \mu + \sqrt{\sigma^2 + \sigma_1^2} \cdot \Phi^{-1}\left(\frac{n-i+1}{n+1}\right). \tag{6}$$

The selected items are top ranked items. They usually have larger values than the median, which means that $i < k < \frac{n+1}{2}$ and $\Phi^{-1}\left(\frac{n-i+1}{n+1}\right) > 0$. Consequently, it is usually true that $E(y_i|\mu, \sigma^2, \sigma_1^2, n) > \mu$.

The distribution of the true value y_i can be obtained through

$$P_{y_i}(x|\mu, \sigma^2, \sigma_1^2, n) = \int_{-\infty}^{\infty} P_{\hat{y}_{i,1}}(z|k, n)\, \phi\left(\frac{x-z}{\sigma_1}\right) \phi\left(\frac{x-\mu}{\sigma}\right) dz. \tag{7}$$

When n is large enough, the distribution of y_i is close to a Gaussian distribution

$$N\left(\mu + \frac{\sigma^2 \cdot \Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}, \frac{\sigma^2 \cdot \sigma_1^2}{\sigma^2+\sigma_1^2} + \frac{i \cdot (n-i+1)}{(n+1)^2 \cdot (n+2) \cdot \left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}\right). \tag{8}$$

So, the expectation of y_i is

$$E(y_i|\mu,\sigma^2,\sigma_1^2,n)=\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}. \tag{9}$$

As the predictor of the second stage is also an unbiased predictor for items values, the expected values from the predictor are equal to the expectation of y_i

$$E\left(\hat{y}_{i,2}|\mu,\sigma^2,\sigma_1^2,n\right)=\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}. \tag{10}$$

Now, the expectation of the difference between predicted values at the first and second stage can be derived from equation (6) and (10), which is

$$\begin{aligned}&E\left(\hat{y}_{i,1}-\hat{y}_{i,2}|\mu,\sigma^2,\sigma_1^2,n\right)=E\left(\hat{y}_{i,1}|\mu,\sigma^2,\sigma_1^2,n\right)-E\left(\hat{y}_{i,2}|\mu,\sigma^2,\sigma_1^2,n\right)=\\&\left(\mu+\sqrt{\sigma^2+\sigma_1^2}\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)-\left(\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}\right)=\frac{\sigma_1^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}.\end{aligned} \tag{11}$$

For the top ranked items, we have $\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)>0$. So, $E\left(\hat{y}_{i,1}-\hat{y}_{i,2}|\mu,\sigma^2,\sigma_1^2,n\right)$ is larger than zero, which means that the average predictions from the first stage predictors tends to be higher than the predictions from the second stage predictors on the final selected subset. As the second stage predictions are unbiased, the predictor of the first stage is actually biased on the selected subset now. This phenomenon is caused by selection process. Indeed, unbiased predictor plus selection process leads to biased predictions on the selected subset.

Let us also briefly discuss about ranking score function. At the *i*th stage we sort items based on predicted value $\hat{y}_{i,j}$. We know that the expected true value is different from $\hat{y}_{i,j}$. Given $\hat{y}_{i,j}$, the expected true value is

$$E\left(y_i|\hat{y}_{i,j},\mu,\sigma^2,\sigma_1^2,n\right)=\frac{\sigma^2\cdot\hat{y}_{i,j}+\sigma_1^2\cdot\mu}{\sigma^2+\sigma_1^2}. \tag{12}$$

Theoretically, we should rank ads through value in Equation (12). However, as this value is monotonically increasing for $\hat{y}_{i,j}$. The larger $\hat{y}_{i,j}$, the larger y_i. So, the ranking and filtering method based on y_i is the same as the method of using the score of $\hat{y}_{i,j}$.

Above conclusion is true for the continuous Gaussian value. We will revisit this for binary event item selection in 2.2.6, of which the model direct estimated value $\hat{y}_{i,j}$ would not be the best score for ranking and filtering items.

2.2.4 System setting optimization

From the expectation of y_i in Equation (9), we can easily see that the total expected value of selected k_1 items at stage one is

$$V_1(S_{k_1}) = k_1 \cdot \mu + \sum_{i=1}^{k_1} \frac{\sigma^2 \cdot \Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}. \tag{13}$$

There are also some other approximations of order statistics [9]. If we adopt one of the approximations instead, total expected reward is then

$$V_1(S_k) = k_1 \cdot \mu + \sum_{i=1}^{k_1 \cdot} \frac{\sigma^2 \cdot \Phi^{-1}\left(\frac{n-i-\alpha+1}{n-2\alpha+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}, \tag{14}$$

where $\alpha = 0.375$. We compare above to approximations through a simulation in Figure 8. The settings in this simulation are $\mu = 0$, $\sigma^2 = 64$, $\sigma_1^2 = 2.5$, $n = 1000$. The one with the adjustment of α is a little bit more accurate.

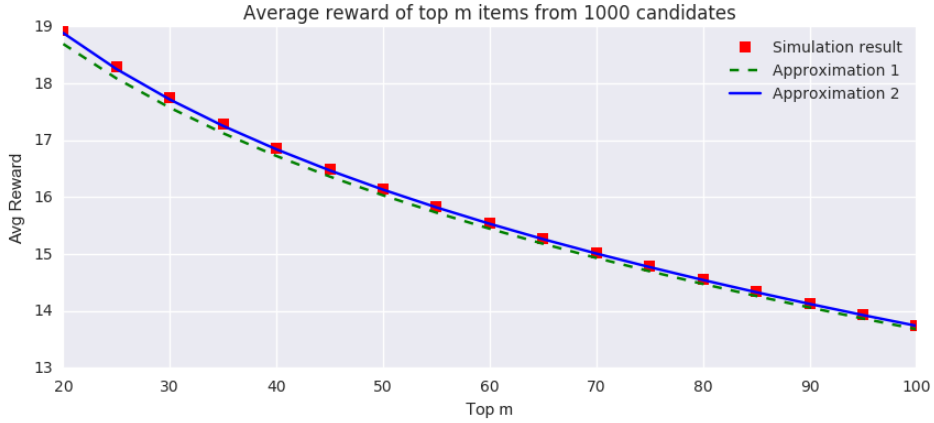


Figure 8. An example of the comparison of the predictions at the first stage, $\mu = 0$, $\sigma^2 = 64$, $\sigma_1^2 = 2.5$, $n = 1000$. Approximation 1 is the curve for equation (13) based on Gaussian assumption in equation (8). Approximation 2 is the curve for equation (14)

It is known that the output from the previous stage is the input for the next stage. If we assume the distribution of y_i is close to a Gaussian distribution, the input of the second stage is a mixture of Gaussian distributions:

$$N\left(\mu + \frac{\sigma^2 \cdot \Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}, \frac{\sigma^2 \cdot \sigma_1^2}{\sigma^2+\sigma_1^2} + \frac{i \cdot (n-i+1)}{(n+1)^2 \cdot (n+2) \cdot \left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}\right), \tag{15}$$

where $i = 1, 2, \dots, k_1$.

If n is very large, this mixture distribution is close to Gaussian distribution $N(\mu', \sigma'^2)$, where μ' and σ'^2 are (see more details in Appendix B)

$$\mu' = \mu + \frac{\sigma^2 \cdot \sum_{i=1}^{k_1} \Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{k_1 \sqrt{\sigma^2 + \sigma_1^2}}, \tag{16}$$

$$\begin{aligned} \sigma'^2 = \frac{\sigma^2 \cdot \sigma_1^2}{\sigma^2 + \sigma_1^2} + \sum_{i=1}^{k_1} \frac{i \cdot (n-i+1)}{(n+1)^2 \cdot (n+2) \cdot k_1 \cdot \left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2} \\ + \sum_{i=1}^{k_1} \left(\mu + \frac{\sigma^2 \cdot \Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2 + \sigma_1^2}}\right)^2 - \left(\frac{\sigma^2 \cdot \sum_{i=1}^{k_1} \Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{k_1 \sqrt{\sigma^2 + \sigma_1^2}}\right)^2. \end{aligned} \tag{17}$$

After using equation (14) iteratively in the second stage, the total expected reward of a two-stage system would be:

$$V_2(S_k) = k \cdot \mu + \sum_{i=1}^{k} \frac{\sigma'^2 \cdot \Phi^{-1}\left(\frac{k_1-i-\alpha+1}{k_1-2\alpha+1}\right)}{\sqrt{\sigma'^2 + \sigma_2^2}}. \tag{18}$$

The maximum value of $V_2(S_k)$ is denoted as $V_{2,max}(S_k)$

$$V_{2,max}(S_k) = \max_{k_1 = k, k+1, \dots n} \left(k \cdot \mu + \sum_{i=1}^{k} \frac{\sigma'^2 \cdot \Phi^{-1}\left(\frac{k_1-i-\alpha+1}{k_1-2\alpha+1}\right)}{\sqrt{\sigma'^2 + \sigma_2^2}} \right). \tag{19}$$

This solution can also be extended to multiple stages. To find the optimal settings, we can exhaustively try different number of stages, and select the one with the largest expected reward as follows

$$V_{max}(S_k) = \max_{j=1,2,..T} \left(V_{j,max}(S_k) \right). \tag{20}$$

2.2.5 Infrastructure resource constraint

The above analysis does not take account of resource cost. In real-world applications, the computational resources are not unlimited. There are many measures of infrastructure cost. Among them, CPU and latency are usually two most important metrics.

The CPU cost relates to model complexity and the number of evaluation times.

$$C^{CPU} = \sum_{j=1}^{T} C_t^{CPU} = \sum_{j=1}^{T} f(\sigma_j^2, k_j) \tag{21}$$

The latency cost also relates to model complexity and the number of evaluation times. Besides these, there is also overhead cost of each stage.

$$C^{Latency} = \sum_{j=1}^{T} C_t^{Latency} = \sum_{j=1}^{T} \left(g\left(\sigma_j^2, k_j\right) + C\right), \tag{22}$$

where C is the latency overhead. You can image that if we have too many stages, the cross-stage communication cost would be nontrivial. Consequently, in real-world applications, engineers usually do not use many stages in cascading selection system. A typical system uses 2~5 stages to ensure that it does not incur significant latency overhead.

After considering CPU and latency, the optimization problem $V_{max}(S_k)$ turns to be

$$\max_{j=1,2,..T} \left(V_{j,max}(S_k)\right) \quad \text{subject to} \quad C^{CPU} < T_1 \text{ and } C^{Latency} < T_2 \tag{23}$$

Figure 9 shows some data points of CPU and latency costs derived from real applications. It is observed that the more predictions, the more cost of CPU and latency. The cost of CPU and latency also increase sharply as predictors get more complex, which is in line with our expectations.

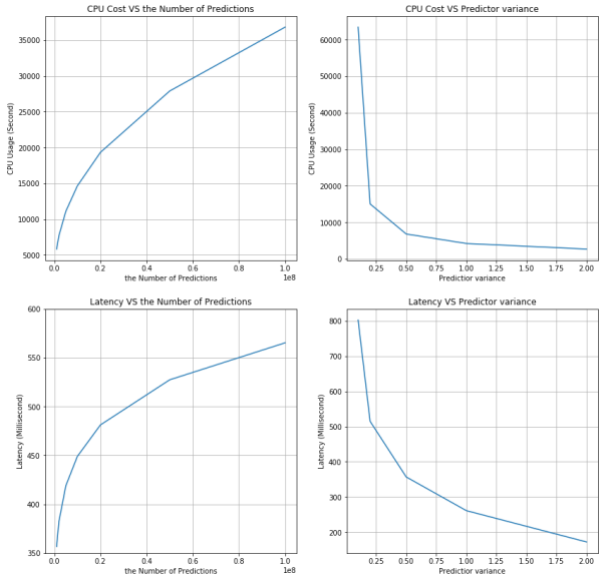


Figure 9. CPU and latency cost curve

2.2.6 Binary event item selection ranking score optimization

We briefly mention in 2.2.2 that there is no difference between directly using predictions and ranking items through equation (12) for continuous reward optimizations. However, the result would be totally different for the binary event prediction problem. It is suboptimal to use model predicted values to rank items for binary event value. The reason is that the prediction uncertainty would be different for each item in binary event problem. We will discuss this problem in this section.

In some applications, the value of each item depends on binary events. Considering ads displaying for example, we need to select ads from a candidate pool for limited slots shown to users. Each ad has a probability turning to be a conversion, which is an action that is counted when a user interacts with the ads, e.g., an online purchase, an app install. Let us assume the conversion rate of each item is p_i, which follows a logit normal distribution $Logitnormal(\mu_p, \sigma_p^2)$. If $\mu_p \ll 1$, $Logitnormal(\mu_p, \sigma_p^2)$ is close to $Lognormal(\mu_p, \sigma_p^2)$.

The values of these conversions are different. The conversion value is represented by b_i which follows a log normal distribution $Lognormal(\mu_b, \sigma_b^2)$. Then the expected value of an item getting displayed is $v_i = p_i \cdot b_i$. We also assume that the value of each item follows a log normal distribution $Lognormal(\mu_v, \sigma_v^2)$.

In real-time bidding platform, the value of conversion, which is equal to bid, could be automatically changed to maintain a reasonable and smooth delivery. For example, if the conversion rate is high, advertiser realizes that this ad does not need a high bid price to win the auction. In this situation, the conversion rates and the expected values are independent of each other. The conditional distribution of p_i given b_i is $Lognormal(\hat{\mu}_p, \hat{\sigma}_p^2)$, where $\hat{\mu}_p$ and $\hat{\sigma}_p^2$ are:

$$\hat{\mu}_p = \mu_p - \frac{\sigma_p^2}{\sigma_p^2+\sigma_v^2}(b_i - \mu_b), \tag{24}$$

$$\hat{\sigma}_p^2 = \frac{\sigma_p^2 \cdot \sigma_v^2}{\sigma_p^2+\sigma_v^2}. \tag{25}$$

However, we do not know the exact conversion rate of each item. Instead, we can only estimate the conversion rate from historical data. Suppose the prediction error of the predictor in stage one follows lognormal distribution $Lognormal(0, \sigma_1^2)$. Then the distribution of predicted value $\hat{p}_i$ is $Lognormal(\hat{\mu}_p, \hat{\sigma}_p^2 + \sigma_1^2)$. The expected value of the true conversion rate given prediction $\hat{p}_i$ and b_i is:

$$E(p_i|\hat{p}_i, b_i) = \text{logit}^{-1}\left(\frac{\frac{\sigma_p^2 \cdot \sigma_v^2}{\sigma_p^2+\sigma_v^2}\text{logit}(\hat{p}_i) + \sigma_1^2 \cdot \hat{\mu}_p}{\frac{\sigma_p^2 \cdot \sigma_v^2}{\sigma_p^2+\sigma_v^2} + \sigma_1^2}\right) = \text{logit}^{-1}\left(\frac{\frac{\sigma_p^2 \cdot \sigma_v^2}{\sigma_p^2+\sigma_v^2}\text{logit}(\hat{p}_i) + \sigma_1^2 \cdot \left(\mu_p - \frac{\sigma_p^2}{\sigma_p^2+\sigma_v^2}(b_i - \mu_b)\right)}{\frac{\sigma_p^2 \cdot \sigma_v^2}{\sigma_p^2+\sigma_v^2} + \sigma_1^2}\right). \tag{26}$$

Now, instead of using model predicted values to rank items, we use $E(p_i|\hat{p}_i, b_i) \cdot b_i$ to sort and filter ads. It is obvious that this method is different from ranking ads through $\hat{p}_i \cdot b_i$.

There are also some cases in which conversion rates and the expected values are not independent of each other. Let us assume that the correlation of bid price and conversion rate is ρ. So, the joint distribution of p_i and b_i is $Lognormal(\mu, \Sigma)$, where the mean μ and covariance Σ are

$$\mu = [\mu_p, \mu_b], \tag{27}$$

$$\Sigma = \begin{bmatrix} \sigma_p^2 & \rho\sigma_p\sigma_b \\ \rho\sigma_p\sigma_b & \sigma_b^2 \end{bmatrix}. \tag{28}$$

It is easy to obtain the joint distribution of observations $\widehat{p_i}$ and $\widehat{b_i}$, which is equal to $Lognormal(\hat{\mu}, \hat{\Sigma})$. Here, the mean $\hat{\mu}$ and covariance $\hat{\Sigma}$ are

$$\hat{\mu} = [\mu_p, \mu_b], \tag{29}$$

$$\hat{\Sigma} = \begin{bmatrix} \sigma_p^2 + \sigma_1^2 & \rho\sigma_b\sqrt{\sigma_p^2 + \sigma_1^2} \\ \rho\sigma_b\sqrt{\sigma_p^2 + \sigma_1^2} & \sigma_b^2 \end{bmatrix}. \tag{30}$$

Then the expected value of the true conversion rate given prediction $\hat{p}_i$ and b_i is:

$$E(p_i|\hat{p}_i, b_i) = \text{logit}^{-1}\left(\frac{(1-\rho^2)\cdot\sigma_p^2\cdot\text{logit}(\hat{p}_i)+\sigma_1^2\cdot\left(\mu_p+\rho\cdot\frac{\sigma_p}{\sigma_b}\cdot(\log(b_i)-\mu_b)\right)}{(1-\rho^2)\cdot\sigma_p^2\cdot+\sigma_1^2}\right). \tag{31}$$

2.3 Experimental results

2.3.1 Accuracy of performance estimation

Our method provides a systematical way to optimize cascading ranking and selection efficiency. But before using it to find the optimization of system setting, we want to verify the accuracy of the performance estimation in equation (18) for different choices of number of stages, and the number of filtered items at each stage.

In our simulation, the candidates are Gaussian distributed with $\mu = 0$, $\sigma^2 = 64$. We select the top 20 items from these 1000 candidate pool, which means $n = 1000$, $k = 20$.

2.3.1.1 Two stages system optimization

Suppose the variance of model prediction at the first stage $\sigma_1^2 = 16$ and the second stage $\sigma_2^2 = 9$. Our theoretical performance estimation is very close to the simulation performance as shown in Figure 10. In this example, the difference between theoretical results (which is calculated through equation (18)) and simulation results is always less than 0.2%. The optimal k_1 we get from our algorithm is 29, which is the same with the results in simulation.

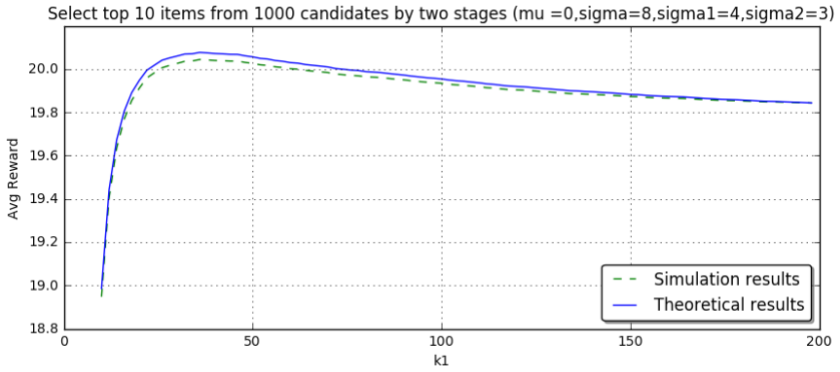


Figure 10. Comparison of simulation results and theoretical results in two-stage system

2.3.1.2 Three stages system optimization

Suppose the variance of model prediction at the first stage $\sigma_1^2 = 25$, the second stage $\sigma_2^2 = 16$, and the third stage $\sigma_2^2 = 9$.

We also plot the comparison of theoretical results and simulation results in Figure 11. It is observed that the difference between theoretical results and simulation results is always less than 1.2%. Although this error is still very small, it is much larger than the difference

of two-stage system. This makes sense as the analysis relies on more assumptions and approximations after adding one extra stage. The comparison for four stages system has not been showed here due to visualization challenge. The difference between our theoretical result and the simulation result is around 3%.

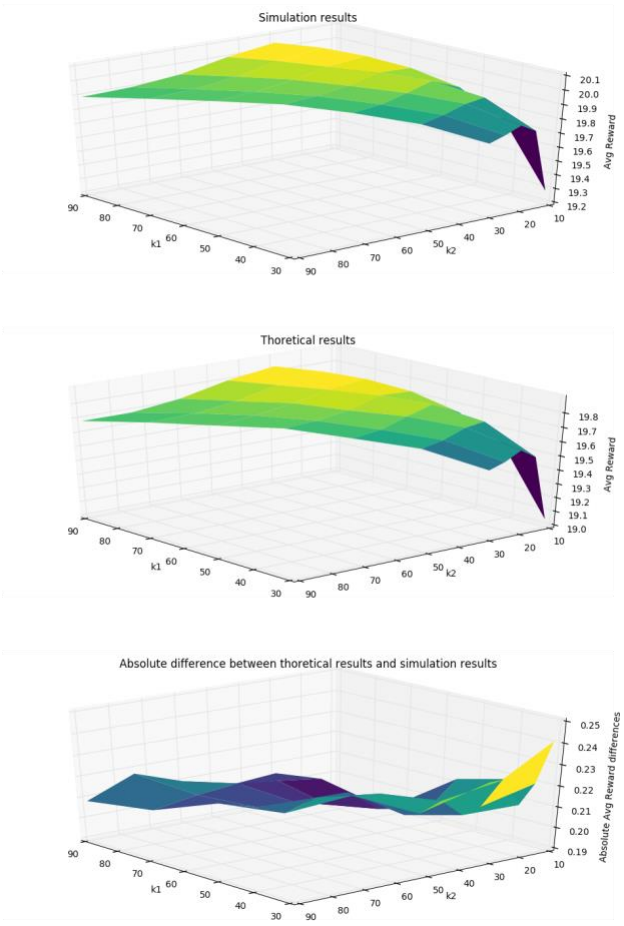


Figure 11. Comparison of simulation results and theoretical results in three-stage system

2.3.2 System setting optimization with infrastructure resource constraints

In real applications, we can add more machines to solve CPU insufficiency. However, adding more computation resources does not help much for latency. So, latency constraint is usually the bound constraint in most cases. In our simulation, we only run the simulation under latency constraint. The simulation result is similar after adding other constraints, e.g.,

CPU constraint. We assume that the latency cost at each stage is $\kappa \frac{k_j^{\alpha}}{\left(\sigma_j^2-\gamma\right)^{\beta}} + C$. Here, $\kappa \in (0,+\infty)$ is the latency cost multiplier. $\alpha \in (0,1)$ is the exponent coefficient for number of ranked items. As predicting more items incurs more latency, the value of α is positive. The latency cost per item decreases as the number of predictions increases due to batch processing. So, the value of α is less than 1.0. For example, if the latency of predicting 10k items is 100ms, the latency of predicting 20k items would be more than 100ms, but usually less than 200ms because the computation resources can be shared more efficiently for 20k item predictions compared with 10k item predictions. $\beta \in (0,+\infty)$ and $\gamma \in (0,+\infty)$ are the parameters for model complexity: β is the exponent coefficient for model complexity, and γ is the lower bound of model variance. Because more complex model, i.e., smaller σ_j^2, incurs more latency, the value of β is also positive. However, σ_j^2 cannot be arbitrarily small. We assume γ is the lower bound. The latency would be positive infinite if σ_j^2 is equal to γ. And $C \in (0,+\infty)$ is the overhead cost for one extra stage. Figure 9 is an example of the real latency cost curve. Summing up the latency cost across all stages, we have the total latency cost $\sum_{j=1}^{T}\left(\kappa \frac{k_j^{\alpha}}{\left(\sigma_j^2-\gamma\right)^{\beta}} + C\right)$.

As there is no other research discussing about how to find the optimal setting as far as we know, we use the single stage system as a baseline in our comparison. In the simulation, the values of these parameters are: $\kappa = 50$, $\alpha = 0.4$, $\beta = 0.6$, $\gamma = 0.7$, and $C = 20$. The total latency budget is 1000ms.

Suppose the problem is to select 10 items from 1000 candidates. Predictor 1 is a simple model. And predictor 2 is a complex model. Here are the lists of the detailed parameters: $\mu = 0, \sigma^2 = 8, \sigma_1^2 = 4, \sigma_2^2 = 1, n = 1000, k = 10.$

Table 2: Average reward in different system settings under infrastructure resource constrains

	Stage 1	**Stage 2**	**Number of items ranked in stage 1**	**Number of items ranked in stage 2**	**Performance: Average reward**
Baseline 1	Predictor 1	NA	1000	NA	18.931
Baseline 2	Predictor 2	NA	293	NA	17.436
Optimal Two-stage solution	Predictor 1	Predictor 2	1000	79	21.010

It is observed that the optimal two-stage solution is much better than two baseline settings in terms of average reward.

2.3.3 Binary event item selection ranking score optimization

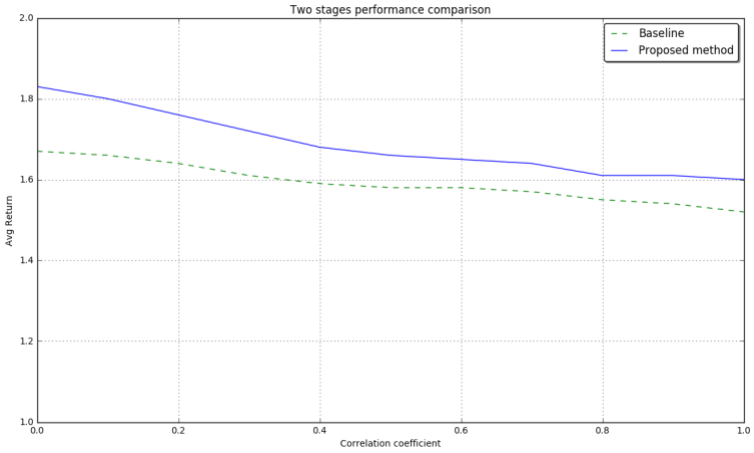
Table 3:Performance comparison of scoring function: average true value*

	Baseline	**Proposed scoring**	**Improvement**
Two Stage, Parameter setting 1	1.62	1.77	9.2%
Two Stage, Parameter setting 2	2.72	3.01	10.7%
Three Stages, Parameter setting	2.14	2.31	7.9%
Three Stages, Parameter setting	3.18	3.67	15.4%
Four Stages, Parameter setting 1	3.28	3.71	15.1%
Four Stages, Parameter setting 2	3.52	4.04	14.8%

*Note: (Parameter setting 1: $\mu_p = -5, \sigma_p^2 = 2, \mu_v = -2, \sigma_v^2 = 3, \sigma_1^2 = 5, , \sigma_2^2 = 4, , \sigma_3^2 = 3, \sigma_4^2 = 2$, Parameter setting 2: $\mu_p = -3, \sigma_p^2 = 1, \mu_v = -1, \sigma_1^2 = 3, , \sigma_2^2 = 2 , \sigma_3^2 = 1, \sigma_4^2 = 0.5$ Two-stage system setting: 1000=>200=>10, Three-stage system setting: 1000=>500=>100=>10, Four-stage system setting: 1000=>500=>200=>50=>10)

We also run a simulation to compare the performance of using model direct predicted value and our proposed score for binary event item selection. Firstly, let us evaluate the performance in the scenario where conversion rates and the expected values are independent to each other. The comparisons include two-stage case, three-stage case, and four-stage case with different parameter settings, which is illustrated in Table 3. The proposed method achieves significant better performances than the baseline method which uses the predicted value from ranking model to rank items directly.

We evaluate the performance of the scenario that conversion rates and the expected values are not independent to each other. Figure 12 shows some performance comparisons given different correlation coefficients. The proposed ranking score function also outperforms baseline method significantly for all two-stage, three-stage, and four-stage problems.



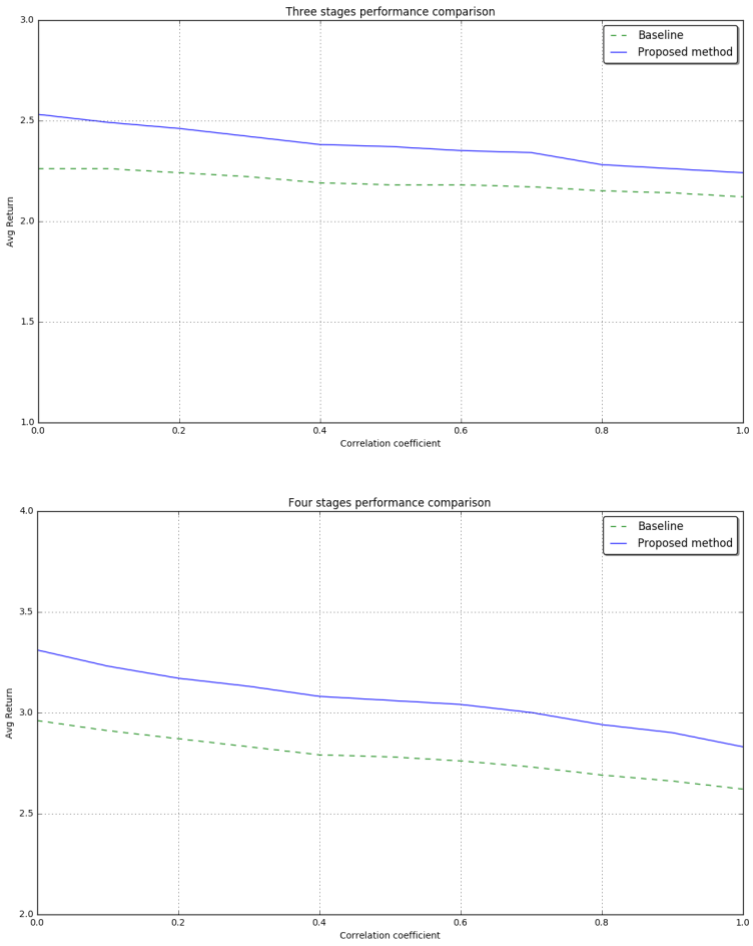


Figure 12. Performance comparison given different correlation coefficients of conversion rate and bid price for two-stage, three-stage, and four-stage problems

CHAPTER III: Explorable item selection

When exploring during item selection process is feasible, the trade-off between exploration and exploitation is the key to obtaining the optimal policy. If the explorable session is unlimited, the problem turns to be a classical multi-armed bandit problem. However, in most situations, exploration is only allowed within a restricted time period. Then the problem turns into a problem of online experiment.

Online experiments are widely used in the advertising industry to compare the performance, e.g., click through rate, conversion rate, of different versions of product features, e.g., ads format, ads category. Some methods, such as A/B testing, and multi-armed bandit algorithms have been studied for decades for online experimental design. However, these methods either ignore potential loss during experiment, or only focus on maximizing the reward in experiment without considering the fact that the high confidence level of the conclusion is important for post-experimental stage usage. The goal of an experiment is to maximize the total reward in two stages: experimental stage and post-experimental stage. Intuitively, it is only worth doing much exploration to obtain a reliable conclusion, that is a result with a high confidence level, if the data volume in post-experimental stage is sufficiently large. In this dissertation, we propose an adaptive solution to optimize the total reward for both the experimental stage and post-experimental stage together. Starting from a simple case, two items and two intervals, we extend the method to two items and multiple intervals, and then to multiple items and multiple intervals. Our algorithms show significant improvements over existing algorithms in different parameters settings of simulations.

3.1 Online experiment introduction

A controlled experiment is one in which everything is held constant except for one variable [10]. Usually a set of data is taken to be a control group. Subsequently, one or more other groups are examined, where all conditions are identical to the control group except this one variable. Sometimes it is necessary to change more than one variable, but all the experimental conditions will be controlled. As a result, only the variables being examined vary between groups. A big advantage of a controlled experiment is the capability of eliminating much of the uncertainty about the experimental results.

Online controlled experiment is widely utilized to make data-driven decisions at Amazon, eBay, Facebook, Google, and many other companies for online advertising and web development [11] [12] [13] [14].

3.2 Related work

3.2.1 A/B testing

The simplest controlled experiment, which has only one group besides control group in test, is called A/B testing [15] [16] [17]. A/B testing is a methodology using randomized experiments with two variants, A and B, which are the “control” and “treatment” in the experiment. It randomly segments users into two groups with equal probability and show two groups different contents. After getting the results, it examines the statistical metrics of two groups and figures out which content has better effect.

A motivating example of using A/B testing is the 2012 United States president campaign [18]. The digital team optimized just about everything from web pages to emails by A/B testing. For example, they lifted conversions by more than 19% by changing the photo on

the splash page, which is shown in Figure 3 in Chapter 1. They overall executed about 500 A/B tests on web pages in a 20-month period, which increased donation conversions by 49% and sign-up conversions by 161%.

A/B testing is an efficient way to capture the effect of the variables concerned. However, this method is defective in terms of accumulating rewards. The reason is that during A/B testing stage, we cannot expose all users to the best version, which results in potential loss of the rewards. If the treatment group is clearly superior, we still have to spend lots of traffic on the control group in order to obtain statistical significance. Let us take an online advertising example for illustration. Suppose true conversion rates of control group and treatment group are 0.10 and 0.12, respectively. The sample size of control group and treatment group are both 10,000. Then, the expected loss during this experiment is 10,000 ·(0.12-0.10) =200 conversions.

3.2.2 Multi-armed bandit problem

The multi-armed bandit (MAB) algorithm has been proposed to substitute A/B testing in industry to alleviate the loss as described in previous section. Google Analytics adopts this method to run online experiments [19]. In probability theory, the multi-armed bandit problem [20] is the problem a gambler faces at a row of slot machines, when deciding which machines to play, how many times to play each machine and in which order to play them. The goal is to maximize the sum of rewards earned through a sequence of lever pulls. This problem is formally equivalent to a one-state Markov Decision Process.

A K-armed bandit problem is defined by rewards $X_i(j)$ for $1 \leq i \leq K$ and $j \geq 1$, where i is the index of arms, and j stands for j_{th} play. when K =2, it becomes a two-armed bandit

problem. For the i_{th} arm, $X_i(j)$ is independent and identical distributed with unknown expectation μ_i. Let T_i(n) be the number of times i_{th} arm has been played during n plays. The objective is finding a policy that chooses the next machine to play based on the sequence of past plays and obtained rewards to maximize the expected cumulative reward.

$$Max \sum_{i=1}^{K} \mu_i E(T_i(n)), \tag{32}$$

where E denotes expectation. It is obvious that if the expectation of each μ_i is known, the maximum reward could be $n\mu^*$, where μ^* is defined as the maximum expectation of all arms. However, these expectations cannot be known (Even though sometimes we have some prior information.). To learn this information, trials need to be taken for each arm, which makes the ideal maximum expected reward unachievable. The total expected loss, which is called regret, after n plays is defined as

$$n\mu^* - \sum_{i=1}^{K} \mu_i E\big(T_i(n)\big)\,. \tag{33}$$

The fundamental tension is between "exploiting" arms which have performed well in the past and "exploring" seemingly inferior arms in case they actually perform even better than other arms.

In the classical paper [21], Lai and Robbins have proved that a policy is the best possible, if it satisfies, for any suboptimal arm i

$$E\big(T_i(n)\big) \leq \Big(\frac{1}{D(p_i||p^*)} + o(1)\Big) \ln n. \tag{34}$$

where $D(p_i||p^*)$ is the Kullback-Leibler divergence between the reward density p_i of i_{th} arm and the reward density p^* of the optimal arm.

A collection of strategies which have been used with multi-armed bandit problems, including ε-greedy, SoftMax, upper confidence bounds (UCB) and Gittins index, are reviewed in the following sections.

3.2.3 *ε*-greedy

ε-greedy algorithm [22] [23] is the most widely used algorithm to solve multi-armed bandit problem. It is quite simple, in both implementation and understanding. And the performance of *ε*-greedy also outperforms many other methods [24].

In the *ε*-greedy algorithm, at each play round j = 1, 2, … the algorithm selects the arm with the highest empirical mean with probability 1- *ε*, and selects a random arm with probability *ε*. Given initial empirical means $\hat{\mu}_i(t)$, for i=1,2…K and t>1, the probability of picking i_{th} arm at time $t+1$ would be

$$p_i(t+1) = \begin{cases} 1-\varepsilon+\varepsilon/\mathrm{K} & if\ i = \arg max_{j=1,..K}\, \hat{\mu}_i(t) \\ \varepsilon/\mathrm{K} & otherwise \end{cases}. \tag{35}$$

A linear bound on the expected regret can be achieved if *ε* is a constant. Actually, any algorithm that involves exploiting and exploring can be considered as *ε*-greedy algorithm with *ε* as a variable. There are some papers that try to improve the method of value setting of *ε* [25][26].

3.2.4 SoftMax

The SoftMax method [28] [29], which is also called Boltzmann exploration, makes the selection using the Boltzmann distribution. Arms with greater empirical means are picked with higher probability. Given initial empirical means $\hat{\mu}_i(t)$, for i=1,2…K and t>1, the probability of picking i_{th} arm at time $t+1$ would be

$$p_i(t+1) = \frac{e^{\hat{\mu}_i(t)/\tau}}{\sum_{j=1}^{K} e^{\hat{\mu}_j(t)/\tau}}, \tag{36}$$

where τ is a temperature parameter to control the randomness of the choice. When τ is close to 0, this algorithm acts like pure greedy selection; when τ is very large, nearly infinity, this algorithm acts like uniform selection.

Linear bound on the expected regret can be achieved if τ is a constant. Though there are some papers trying to improve the method that sets the value of τ [25], the practical advantage of using a constant value is not obvious according to experiments [27].

3.2.5 UCB

The Upper Confidence Bound (UCB) family of algorithms, which use confidence bounds to deal with an exploitation-exploration trade-off, has been proposed in literature [26] [30] [31].

The policy UCB1 is the simplest UCB algorithm, which is derived from the index-based policy [32]. It maintains the number of times that each arm has been played, denoted by $n_i(t)$, in addition to the empirical means. Initially, each arm is played once. Afterwards, at round *t*, the algorithm greedily picks the arm $i^*(t)$ as follows:

$$i^*(t) = \arg \underset{i = 1, \ldots K}{Max} \hat{\mu}_i(t) + \sqrt{\frac{2 \ln t}{n_i(t)}}, \tag{37}$$

The sum contains two items. "The first item $\hat{\mu}_i(t)$ is simply current empirical means of reward. The second item $\sqrt{\frac{2 \ln t}{n_i(t)}}$ is related to the size of one-sided confidence interval for

the average reward within which the true expected reward falls with overwhelming probability." It is proved that at each turn *t*, the expected regret of UCB1 is bounded by [25]

$$8 \sum_{i:\mu_i<\mu^*} \frac{\ln(t)}{\mu^* - \mu_i} + (1 + \frac{\pi^2}{3}) \sum_{i=1}^{K} (\mu^* - \mu_i). \tag{38}$$

UCB1 achieves the optimal regret up to a multiplicative constant because the expected regret matches logarithmical bound [25].

Subsequent UCB algorithms, which go beyond UCB1 algorithm, have been developed to improve efficiency in solving multi-armed bandit problem [33] [34] [35].

3.2.6 Gittins index

An alternative formulation of the bandit problem has been studied [36] [37] [38] [39], which is called Gittins index. This algorithm does not have to trade off directly between exploration and exploitation. It assumes a geometrically discounted stream of future rewards with present value. The true reward distribution assumed to be known through a process of forward induction. Gittins provides a back-induction algorithm for computing Gittins index, which is the expected discounted present value of playing an arm, assuming optimal play in the future. Thus, by definition, playing the arm with the largest Gittins index will maximize the expected present value of discounted future rewards.

Calculating Gittins indices exactly is extremely time consuming [40]. There are some approximation calculation methods [41] [42] to accelerate calculation.

Gittins index is an inconsistent estimator of the location of the optimal arm. The Gittins policy eventually chooses one arm on which to continue forever, and there is a positive probability that the chosen arm is sub-optimal [43].

3.3 Problem Description

Let us take advertising again as an example. Assume that we have an ad displaying spot, and there are several ad candidates which can be chosen to fill in the spot. We do not know the click through rates (CTR) of these ads, which have to be estimated through experiments. After the experiments, we will decide to choose one ad to be displayed. Our goal is to maximize the total ad clicks during the combined experiment and post-experimental stages.

Let index i and t denote item index and time interval index in experiment respectively, $i = 1, 2, \ldots, m$ and $t = 0, 1, 2, \ldots, k\text{-}2$. N_t is the total number of impressions during interval t. Let p_i denote the unobserved click through rate (CTR) of item i, which needs to be explored through the experiment. The time interval k-1 is taken as post-experimental stage. And N_{k-1} is usually much larger than N_t, where $t < k\text{-}1$. Our goal is to find an optimal policy π of allocating traffic at each time interval during the experiment to maximize the total ads clicks in both experimental and post-experimental stage. We use $r_{i,t}$ to represent allocation ratio, which is the fraction of displays allocated to item i during time interval t. The value of p_i can be estimated from $c_{i,t}$ clicks from $r_{i,t} \cdot N_t$ impressions. As the observed information: number of clicks $c_{i,t}$ follows Binomial($r_{i,t} \cdot N_t$, p_i), we assume that prior follows a Beta(α_t, β_t). Combined with observed information during time interval t, posterior p_i is Beta($\alpha_t + c_{i,t}$, $\beta_t + r_{i,t} \cdot N_t - c_{i,t}$), which is also the prior at time t+1.

3.3.1 Two items and two intervals (2×2 Case)

We start our analysis from two items and two intervals. We can interpret the first interval, interval 0, as experimental stage and the second interval, interval 1, as post-experimental stage. Below we recap the notations.

N_0 , N_1: the expected number of impressions during time interval 0 and 1.

p_1, p_2: true click through rates of item 1 and 2.

($\alpha_{1,0}$, $\beta_{1,0}$), ($\alpha_{2,0}$, $\beta_{2,0}$): prior parameters of item 1 and 2 at time 0.

($\alpha_{1,1}$, $\beta_{1,1}$), ($\alpha_{2,1}$, $\beta_{2,1}$): prior parameters of item 1 and 2 at time 1.

$r_{1,0}$, $r_{2,0}$: traffic allocation ratios for item 1 and 2 at time 0.

$r_{1,1}$, $r_{2,1}$: traffic allocation ratios for item 1 and 2 at time 1.

$c_{1,0}$, $c_{2,0}$: number of clicks of item 1 and 2 during time interval 0.

$c_{1,1}$, $c_{2,1}$: number of clicks of item 1 and 2 during time interval 1.

The sum of ratios at time interval 0 and 1 should both be 1:

$$r_{1,0} + r_{2,0} = 1, \tag{39}$$

$$r_{1,1} + r_{2,1} = 1. \tag{40}$$

So, we only need to find out the optimal $r_{1,0}$ and $r_{1,1}$.

The expected total numbers of clicks during time interval 0 and 1 are given by

$$E_0 = r_{1,0} \cdot N_0 \cdot E(p_1|\alpha_{1,0}, \beta_{1,0}) + (1 - r_{1,0}) \cdot N_0 \cdot E(p_2|\alpha_{2,0}, \beta_{2,0}), \tag{41}$$

$$E_1 = r_{1,1} \cdot N_1 \cdot E(p_1|\alpha_{1,1}, \beta_{1,1}) + (1 - r_{1,1}) \cdot N_1 \cdot E(p_2|\alpha_{2,1}, \beta_{2,1}). \tag{42}$$

The goal is to find $r_{1,0}$ and $r_{1,1}$ to maximize the total expected number of clicks in the two intervals given by

$$\max_{r_{1,0}, r_{1,1}} (E_0 + E_1) = \max_{r_{1,0}} \left(E_0 + \max_{r_{1,1}} E_1 \right). \tag{43}$$

For the sub-problem $\max_{r_{1,1}} E_1$, the solution is obviously:

$$r_{1,1}^* = \begin{cases} 1, & if\ E(p_1|\alpha_{1,1}, \beta_{1,1}) > E(p_2|\alpha_{2,1}, \beta_{2,1}) \\ 0, & otherwise \end{cases}. \tag{44}$$

Then the objective function in equation (44) turns to be

$$\begin{aligned} \max_{r_{1,0}} \Big(& r_{1,0} \cdot N_0 \cdot E\left(p_1 \middle| \alpha_{1,0}, \beta_{1,0}\right) + (1 - r_{1,0}) \cdot N_0 \cdot E\left(p_2 \middle| \alpha_{2,0}, \beta_{2,0}\right) + \\ & E\left(q \cdot N_1 \cdot p_1 + (1 - q) \cdot N_1 \cdot p_2 \middle| \alpha_{1,0}, \beta_{1,0}, \alpha_{2,0}, \beta_{2,0}, r_{1,0}, N_0\right)\Big). \end{aligned} \tag{45}$$

Here, q is the probability that the expected click rate of item1 is larger than the expected click rates of item 2 at time 1, which is

$$\Pr\left(\frac{\alpha_{1,0} + c_{1,0}}{\alpha_{1,0} + \beta_{1,0} + r_{1,0} \cdot N_0} > \frac{\alpha_{2,0} + c_{2,0}}{\alpha_{2,0} + \beta_{2,0} + (1 - r_{1,0}) \cdot N_0} \middle| p_1, p_2, r_{1,0}, N_0 \right). \tag{46}$$

As $c_{1,0}$ and $c_{2,0}$ are two independent binomials given p_1 and p_2, q is equal to

$$\sum_{x=0}^{(1-r_{1,0})\cdot N_0}\left(\binom{(1-r_{1,0})\cdot N_0}{x}\cdot {p_2}^{x}\cdot(1-p_2)^{(1-r_{1,0})\cdot N_0-x}\cdot \sum_{y=m(x)}^{r_{1,0}\cdot N_0}\binom{r_{1,0}\cdot N_0}{y}{p_1}^{y}\cdot(1-p_1)^{r_{1,0}\cdot N_0-y}\right), \tag{47}$$

where $m(x)$ is $\left\lceil \frac{\alpha_{1,0}+\beta_{1,0}+r_{1,0}\cdot N_0}{\alpha_{2,0}+\beta_{2,0}+(1-r_{1,0})\cdot N_0}\left(\alpha_{2,0}+x\right)-\alpha_{1,0}\right\rceil$.

The complexity of calculating q depends on $(1\text{-}r_{1,0})\cdot N_0$ and $r_{1,0}\cdot N_0$. The larger these two terms are, the more complex the calculation would be. Could we find a cheaper way of calculating q? The answer is yes. As we know, when $(1\text{-}r_{1,0})\cdot N_0$ and $r_{1,0}\cdot N_0$ are large enough, $c_{1,0}$ and $c_{2,0}$ are close to normal distribution, which makes q approximately to be

$$\phi\left(\frac{\frac{\alpha_{1,0}+r_{1,0}\cdot N_0\cdot p_1}{\alpha_{1,0}+\beta_{1,0}+r_{1,0}\cdot N_0}-\frac{\alpha_{2,0}+(1-r_{1,0})\cdot N_0\cdot p_2}{\alpha_{2,0}+\beta_{2,0}+(1-r_{1,0})\cdot N_0}}{\sqrt{\frac{r_{1,0}\cdot N_0\cdot p_1\cdot(1-p_1)}{\left(\alpha_{1,0}+\beta_{1,0}+r_{1,0}\cdot N_0\right)^2}+\frac{(1-r_{1,0})\cdot N_0\cdot p_2\cdot(1-p_2)}{\left(\alpha_{2,0}+\beta_{2,0}+(1-r_{1,0})\cdot N_0\right)^2}}}\right), \tag{48}$$

where Φ is the cumulative density function of normal distribution.

3.3.2 Two items and multi intervals (2×k)

In real-world applications, we can usually adjust allocation ratios during the experiment. Suppose there are totally k batches (time intervals). The problem would be:

$$\max_{r_{1,i},i=0,1,\ldots,k-1}\sum_{i=0}^{i=k-1}E_i. \tag{49}$$

The complexity of solving this problem is very high. To make it simple, we break the problem into a sequential 2×2 sub-problems: the first stage includes the current batch; the second stage includes all the remaining batches. The algorithm is shown in Figure 13.

However, this method has a problem if the total experiment size is large, but each "batch" is too small. In this situation, the 1st stage of this method always has very few samples. So, in each sub-problem, the allocation decision for the 2nd batch is primarily decided by the prior information. The result of the "first" stage experiment does not make much impact on the allocation policy, which would lead this method to a greedy algorithm. The performance of greedy algorithm is bad if the size of post-experiment is large.

Randomized Policy: Sequential Two Stages Method 1

Initialization: Initialize $\alpha_{1,0}, \beta_{1,0}, \alpha_{2,0}, \beta_{2,0}$ based on prior information.

Loop: for each batch i, i = 0, 1, 2,…, k-1

The allocation probability $r_{1,i}$ is calculated as

$$\max_{r_{1,i}} \left(r_{1,i} \cdot N_i \cdot E\left(p_1 \middle| \alpha_{1,i}, \beta_{1,i}\right) + \left(1 - r_{1,i}\right) \cdot N_i \cdot E\left(p_2 \middle| \alpha_{2,i}, \beta_{2,i}\right) + E\left(q \cdot \sum_{j=i+1}^{k-1} N_j \cdot p_1 + (1-q) \cdot \sum_{j=i+1}^{k-1} N_j \cdot p_2 \middle| \alpha_{1,i}, \beta_{1,i}, \alpha_{2,i}, \beta_{2,i}, r_{1,i}\right)\right).$$

q is equal to

$$\sum_{x=0}^{(1-r_{1,i})\cdot N_i} \left(\binom{(1-r_{1,i})\cdot N_i}{x} \cdot {p_2}^x \cdot (1-p_2)^{(1-r_{1,i})\cdot N_i - x} \cdot \sum_{y=m(x)}^{r_{1,i}\cdot N_i} \binom{r_{1,i}\cdot N_i}{y} {p_1}^y \cdot (1-p_1)^{r_{1,i}\cdot N_i - y} \right),$$

where $m(x)$ is $\left\lceil \frac{\alpha_{1,i}+\beta_{1,i}+r_{1,i}\cdot N_i}{\alpha_{2,i}+\beta_{2,i}+(1-r_{1,i})\cdot N_i}(\alpha_{2,i}+x) - \alpha_{1,i} \right\rceil$.

Select user into control group with probability $r_{1,i}$. Select user into treatment group with $(1 - r_{1,i})$.

Update Parameters:

$$\alpha_{1,i+1} = \alpha_{1,i} + c_{1,i},$$
$$\beta_{1,i+1} = \beta_{1,i} + n_{1,i} - c_{1,i},$$
$$\alpha_{2,i+1} = \alpha_{2,i} + c_{2,i},$$
$$\beta_{2,i+1} = \beta_{2,i} + n_{2,i} - c_{2,i}.$$

To solve this problem, we consider another strategy. Take the first sub-problem for example: here we still consider dividing the whole problem into two batches: not only interval 0 and interval 1~k-1, but also consider other dividing options: interval 0~l and interval l+1 ~ k-1, which l = 0, 1, …, k-1. Choose the one that has the largest expected clicks to be the final solution of sub-problem 1. The calculated optimal ratio will only be

used in the following time interval. At the next time interval, we will collect new results and re-calculate the optimal ratio. The algorithm is shown in Figure 14.

Randomized Policy: Sequential Two Stages Method 2

Initialization: Initialize $\alpha_{1,0}, \beta_{1,0}, \alpha_{2,0}, \beta_{2,0}$ based on prior information.

Loop: for each batch i, $i = 0, 1, 2, \ldots, k\text{-}1$

The allocation probability $r_{1,i}$ is calculated as

$$\max_{r_{1,i}, l_i} \left(r_{1,i} \cdot \sum_{j=i}^{l_i} N_j \cdot E\left(p_1 \middle| \alpha_{1,i}, \beta_{1,i}\right) + \left(1 - r_{1,i}\right) \cdot \sum_{j=i}^{l_i} N_j \cdot E\left(p_2 \middle| \alpha_{2,i}, \beta_{2,i}\right) + E\left(q \cdot \sum_{j=l_i+1}^{k-1} N_j \cdot p_1 + (1-q) \cdot \sum_{j=l_i+1}^{k-1} N_j \cdot p_2 \middle| \alpha_{1,i}, \beta_{1,i}, \alpha_{2,i}, \beta_{2,i}, r_{1,i}\right)\right).$$

q is equal to

$$\sum_{x=0}^{(1-r_{1,i}) \cdot \sum_{j=i}^{l_i} N_j} \left(\binom{(1-r_{1,i}) \cdot \sum_{j=i}^{l_i} N_j}{x} \cdot {p_2}^x \cdot (1-p_2)^{(1-r_{1,i}) \cdot \sum_{j=i}^{l_i} N_j - x} \cdot \sum_{y=m(x)}^{r_{1,i} \cdot N_i} \binom{r_{1,i} \cdot \sum_{j=i}^{l_i} N_j}{y} {p_1}^y \cdot (1-p_1)^{r_{1,i} \cdot \sum_{j=i}^{l_i} N_j - y} \right),$$

where $m(x)$ is $\left\lceil \frac{\alpha_{1,i} + \beta_{1,i} + r_{1,i} \cdot N_i}{\alpha_{2,i} + \beta_{2,i} + (1-r_{1,i}) \cdot N_i} \left(\alpha_{2,i} + x\right) - \alpha_{1,i} \right\rceil$.

Select users into control group with probability $r_{1,i}$, and user into treatment group with $(1 - r_{1,i})$.

Update Parameters:

$$\alpha_{1,i+1} = \alpha_{1,i} + c_{1,i},$$
$$\beta_{1,i+1} = \beta_{1,i} + n_{1,i} - c_{1,i},$$
$$\alpha_{2,i+1} = \alpha_{2,i} + c_{2,i},$$
$$\beta_{2,i+1} = \beta_{2,i} + n_{2,i} - c_{2,i}.$$

Figure 13. Sequential Two Stages Method

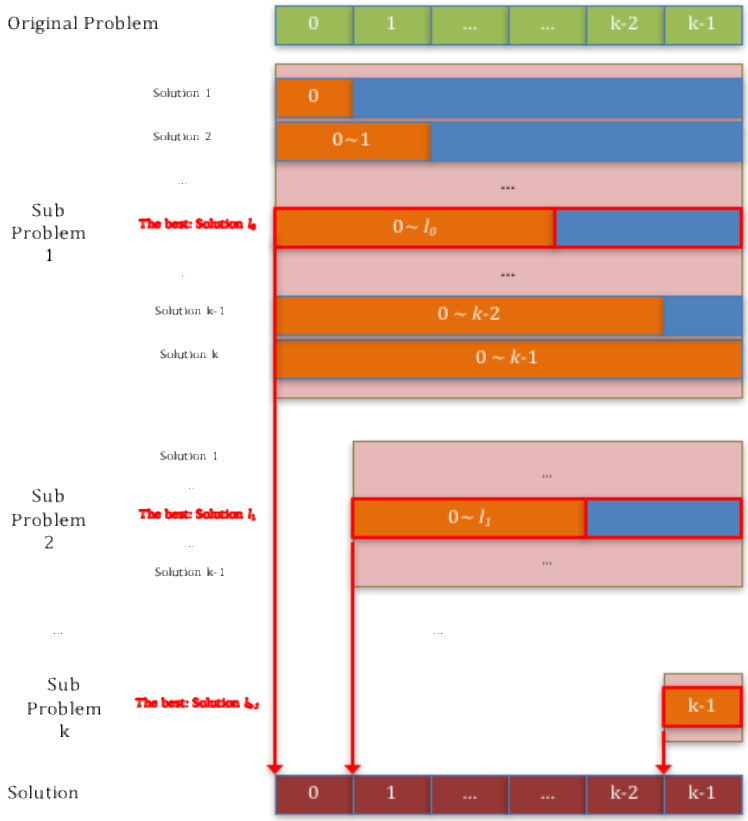


Figure 14. Sequential Two Stages Method 2

3.3.3 Multi items and multi intervals (*m*×*k*)

The calculation is very complex for multiple items, even with only two intervals, not to mention multiple intervals. We need to estimate pairwise differences among items to select promising ones. To simplify the calculation and get a scalable solution, we propose an algorithm which firstly leverages the proposed method 2 to calculate pairwise ratios of each pairs, e.g., $r_{p,i}^{(q)}$ and $r_{q,i}^{(p)}$ for the pair item p and q, and then normalizes the ratio for each item. Though it is not the theoretical optimal solution, it can achieve a great performance, which is shown in our experiments. The detail of this method is as follow:

Randomized Policy: Multi items and multi intervals
Initialization: Initialize $\alpha_{s,0}, \beta_{s,0}$, s = 0, 1, 2,…, m based on prior information.
Loop: for each batch i, i = 0, 1, 2,…, k-1
Loop: for each item pair: p and q, calculate $r_{p,i}^{(q)}$ and $r_{q,i}^{(p)}$ through method 2
Select each item with the normalized probability, e.g., item s

$$\frac{2\cdot\sum_{s=1,q\neq s}^{m} r_{s,i}^{(q)}}{k\cdot(k-1)}.$$

Update parameters for each $\alpha_{s,i+1}, \beta_{s,i+1}$

3.4 Experimental results

3.4.1 Experiment design

In the experiment, data of control group and treatment group are randomly generated given corresponding reward rate. One way of simulation is to generate data at sampling stage: at each time period, data are generated given reward rate and sampling size derived from different algorithms, as Figure 15 shows. However, in that case there is a problem that the data generated for different algorithms is not the same with each other, which adds more uncertainty of the experimental results.

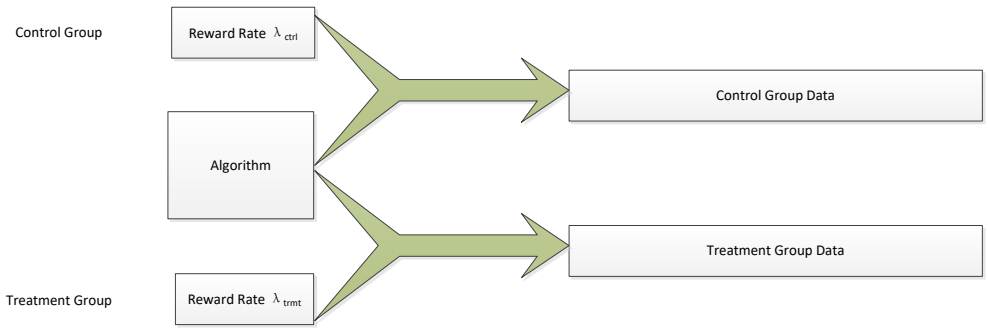


Figure 15. Scheme 1: Data is generated at algorithm running stage

To address this issue, another simulation scheme is adopted: two groups of sequential data are generated ahead of running the algorithms. Each algorithm fetches data according to its own sampling strategy. Now, all the algorithms play on the same data, which is fairer

for performance comparison. Of course, we can obtain more accurate evaluations for these algorithms in this way. Figure 16 shows how this scheme works.

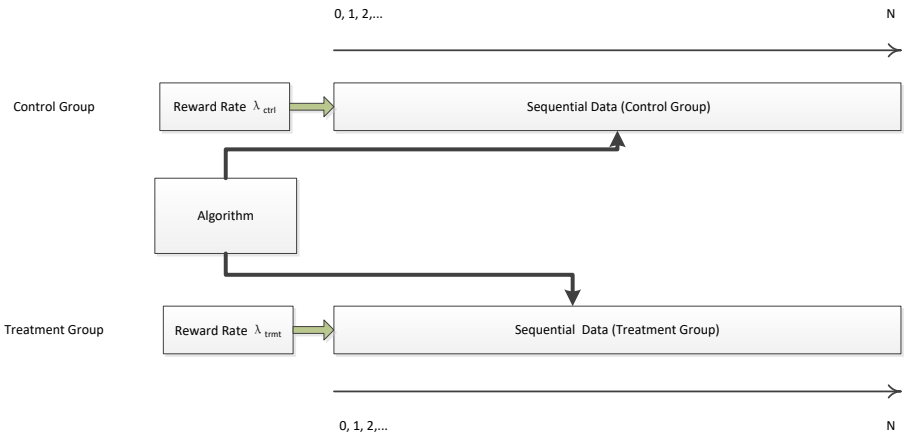


Figure 16. Scheme 2: Data is generated ahead of algorithm running

3.4.2 Experiment setup

In this study, for each algorithm, we run the simulation with the same settings for 100 times to obtain the average performance and the deviation of the performance. At each time, the conversion rates of two groups are generated from a uniform distribution U(0, 0.1). The total number of each time interval follows a Poisson distribution Poisson(100).

Experiments run with two different lengths of time periods, 5 and 100, respectively to represent situations with a limited number of samples and adequate number of samples. It also runs in two different situations: 1. There is some prior information of the control group. 2. There is no prior information of the control group. For the 5 time periods experiment, 250 observations are assigned. For the 100 time periods experiment, 5000 observations are assigned. No prior information of treatment group is assigned in the experiment. Future

period F is set to be 0, 100 and 500 for 5 time periods experiment and 0, 1000, 5000 for 100 time periods experiment. Totally, experiment has 12 different settings as Table 4 shows.

The evaluated algorithms include equal probability allocation (EPA), which equally allocate the traffic to two groups at each time interval, equal size allocation (ESA), which makes sure that the observation sizes, including both prior information and observation, of two groups are equal, pure greedy, UCB1, randomized probability matching (RPM), and our proposed: sequential two stages (STS) method 2.

Table 4: Experiments Setup Table for Virtual Future Measure Experiment

Experiment ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Prior Information Size	0	0	0	250	250	250	0	0	0	5000	5000	5000
Time Period Length of Experimental Stage	5	5	5	5	5	5	100	100	100	100	100	100
Future period of Post-Experimental Stage	0	100	500	0	100	500	0	1000	5000	0	1000	5000

3.4.3 Two items multiple stages (2×*k*)

Here are the simulation results for two items multiple stages problem.

Time Period Length of Experimental Stage = 5, Prior Information Size = 0

Table 5: Mean and SD of regrets (Time Period Length of Experimental Stage = 5, Prior Information Size = 0)

Future period of Post-Experimental Stage		Equal Probability Allocation (EPA)	Equal Size Allocation (ESA)	Greedy	UCB1	Randomized Probability Matching (RPM)	Sequential two stages (STS)

0	Mean	7.410283	7.413877	4.024992	6.322252	3.212737	2.597022
	SD	0.00418038	0.00195284	0.0120176	0.00252667	0.00349289	0.00290052
100	Mean	13.37419	13.1108	26.03839	12.59857	13.37689	11.05450
	SD	0.109556	0.0522560	0.418003	0.113215	0.144779	0.124510
500	Mean	37.74059	36.33763	118.0026	37.99229	53.65423	36.11438
	SD	0.500524	0.263101	2.20135	0.621494	0.705577	0.656187

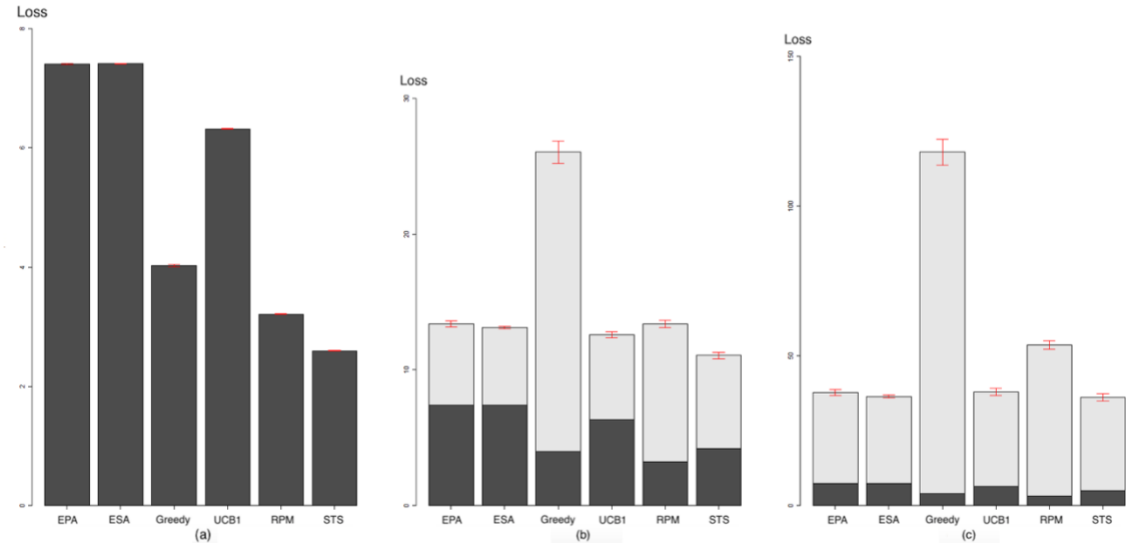


Figure 17. Cumulative regrets (Time Period Length of Experimental Stage = 5, Prior Information Size = 0) under **(a)** Future period = 0 **(b)** Future period = 100 **(c)** Future period = 500. Note that black bar is cumulative regrets during experiments stage and grey bar is cumulative virtual future regrets.

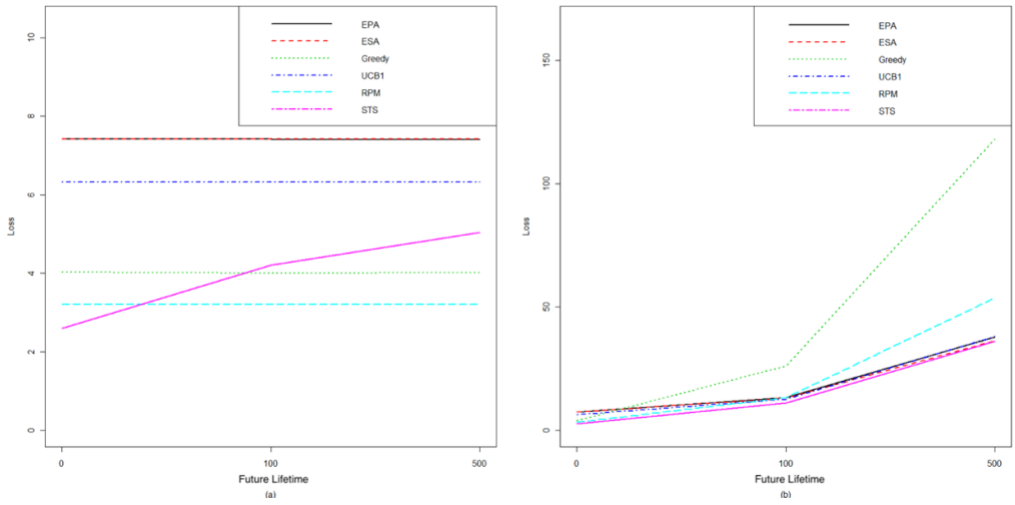


Figure 18. How does Future period change regrets (Time Period Length of Experimental Stage = 5, Prior Information Size = 0) (a) Cumulative regrets during experiments stage (b) Total cumulative regrets

Time Period Length of Experimental Stage = 5, Prior Information Size = 250

Table 6: Mean and SD of regrets (Time Period Length of Experimental Stage = 5, Prior Information Size = 250)

Future period of Post-Experimental Stage		Equal Probability Allocation (EPA)	Equal Size Allocation (ESA)	Greedy	UCB1	Randomized Probability Matching (RPM)	Sequential two stages (STS)
0	Mean	7.413597	7.81741	3.086902	5.649278	2.447000	2.111957
	SD	0.00446910	0.00168245	0.00588717	0.00258550	0.00290916	0.00273641
100	Mean	11.84425	11.35162	17.00135	9.412254	9.145567	7.366655
	SD	0.0464996	0.0361032	0.203317	0.0769668	0.144690	0.0897081
500	Mean	29.88097	25.40701	72.06019	24.80907	37.4481	23.15598
	SD	0.294549	0.110346	1.162603	0.37335	0.854644	0.370404

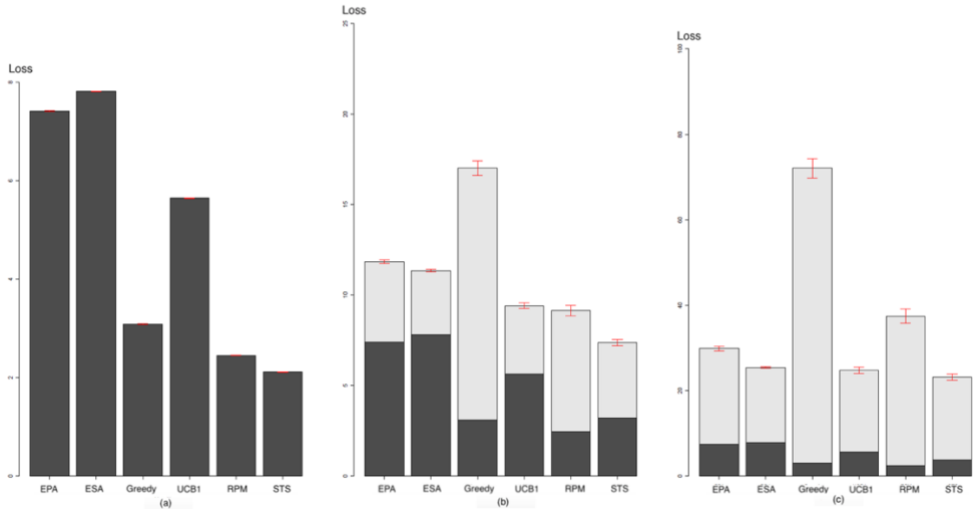


Figure 19. Cumulative regrets (Time Period Length of Experimental Stage = 5, Prior Information Size = 250) under **(a)** Future period = 0 **(b)** Future period = 100 **(c)** Future period = 500. Note that black bar is cumulative regrets during experiments stage and grey bar is cumulative virtual future regrets.

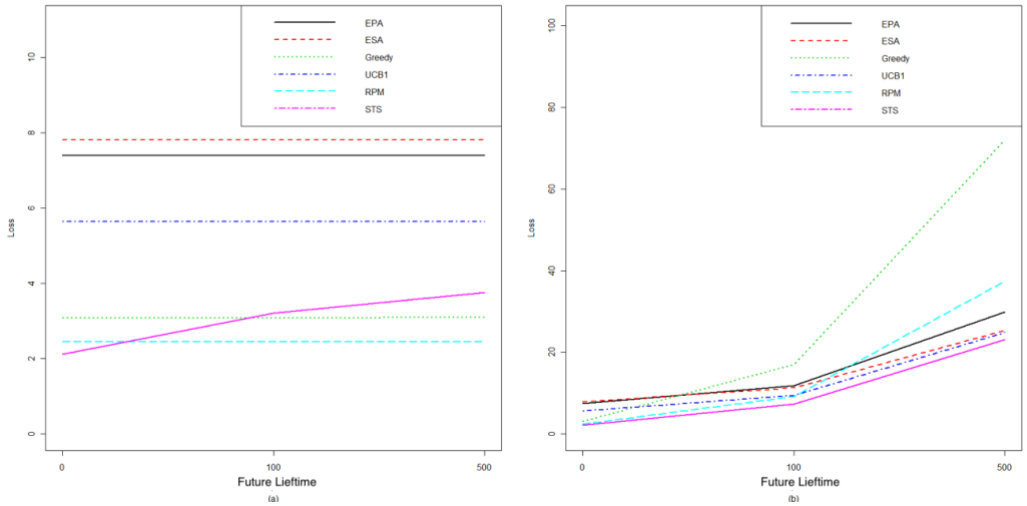


Figure 20. How does Future period change regrets (Time Period Length of Experimental Stage = 5, Prior Information Size = 250) (a) Cumulative regrets during experiments stage (b) Total cumulative regrets

Time Period Length of Experimental Stage = 100, Prior Information Size = 0

Table 7: Mean and SD of regrets (Time Period Length of Experimental Stage = 100, Prior Information Size = 0)

Future period of Post-Experimental Stage		Equal Probability Allocation (EPA)	Equal Size Allocation (ESA)	Greedy	UCB1	Randomized Probability Matching (RPM)	Sequential two stages (STS)
0	Mean	148.1488	148.1653	24.96327	65.17191	9.761653	9.613503
	SD	0.0182676	0.00206033	0.399701	0.00208211	0.0821495	0.00243636
1000	Mean	149.5394	149.5519	230.3115	66.55326	47.35902	17.67864
	SD	0.0185711	0.00196484	4.645897	0.00226715	0.979769	0.00224975
5000	Mean	155.0736	155.0955	1068.077	72.09497	188.1976	26.51378
	SD	0.0169755	0.00208992	22.25767	0.00214160	4.49216	0.00299169

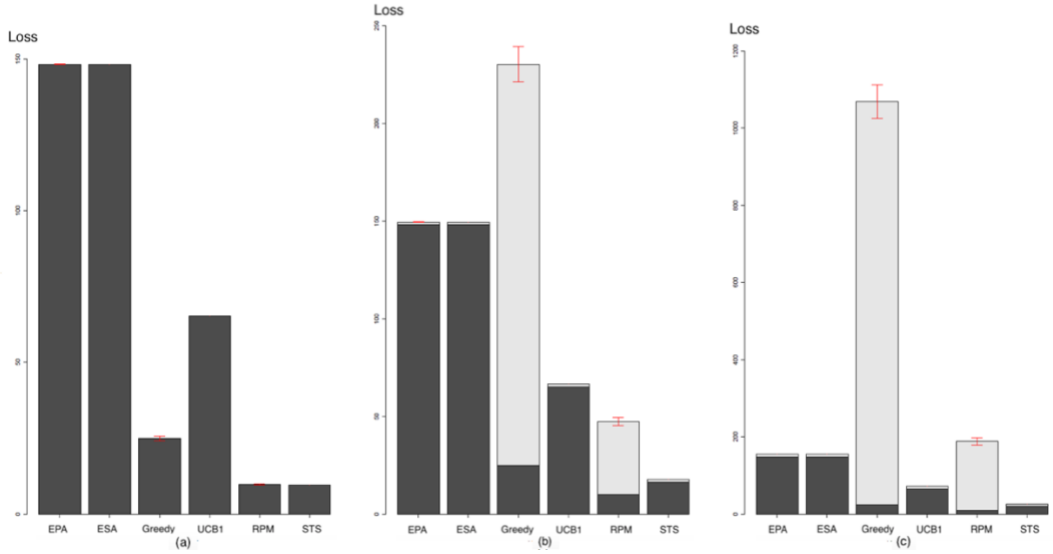


Figure 21. Cumulative regrets (Time Period Length of Experimental Stage = 100, Prior Information Size = 0) under **(a)** Future period = 0 **(b)** Future period = 1000 **(c)** Future period = 5000. Note that black bar is cumulative regrets during experiments stage and grey bar is cumulative virtual future regrets.

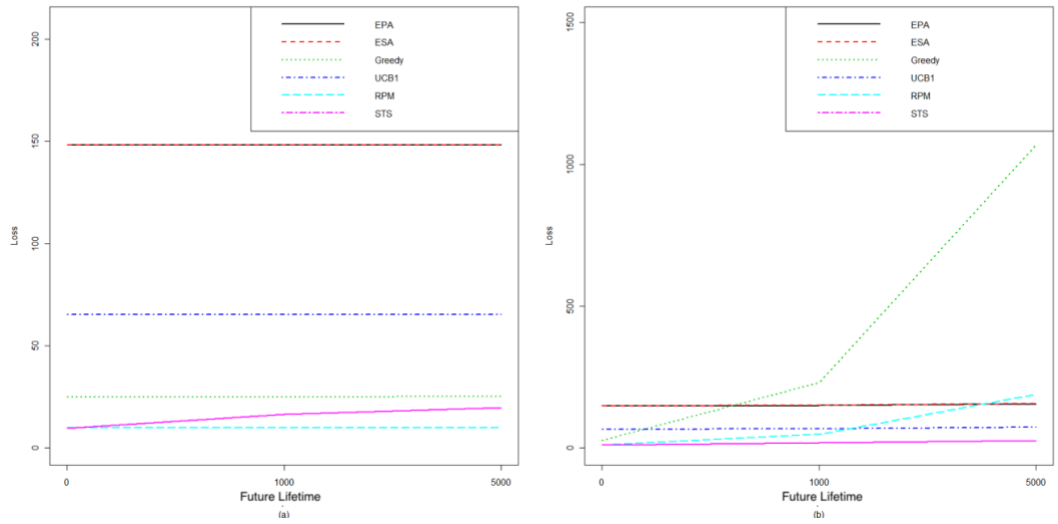


Figure 22. How does Future period change regrets (Time Period Length of Experimental Stage = 100, Prior Information Size = 0) (a) Cumulative regrets during experiments stage (b) Total cumulative regrets

Time Period Length of Experimental Stage = 100, Prior Information Size = 5000

Table 8: Mean and SD of regrets (Time Period Length of Experimental Stage = 100, Prior Information Size = 5000)

Future period of Post-Experimental Stage		Equal Probability Allocation (EPA)	Equal Size Allocation (ESA)	Greedy	UCB1	Randomized Probability Matching (RPM)	Sequential two stages (STS)

0	Mean	148.1904	149.0989	21.56430	42.21271	9.422874	5.280005
	SD	0.0189748	0.00194009	0.376613	0.00275169	0.0176001	0.00297323
1000	Mean	150.5852	149.4824	252.4254	43.6029	45.78592	8.344196
	SD	0.0160058	0.00180474	2.459249	0.00246558	0.218074	0.00292461
5000	Mean	157.1022	155.0245	1172.329	49.13693	191.1284	15.85606
	SD	0.0172936	0.00174546	10.61887	0.00330607	1.018030	0.0036232

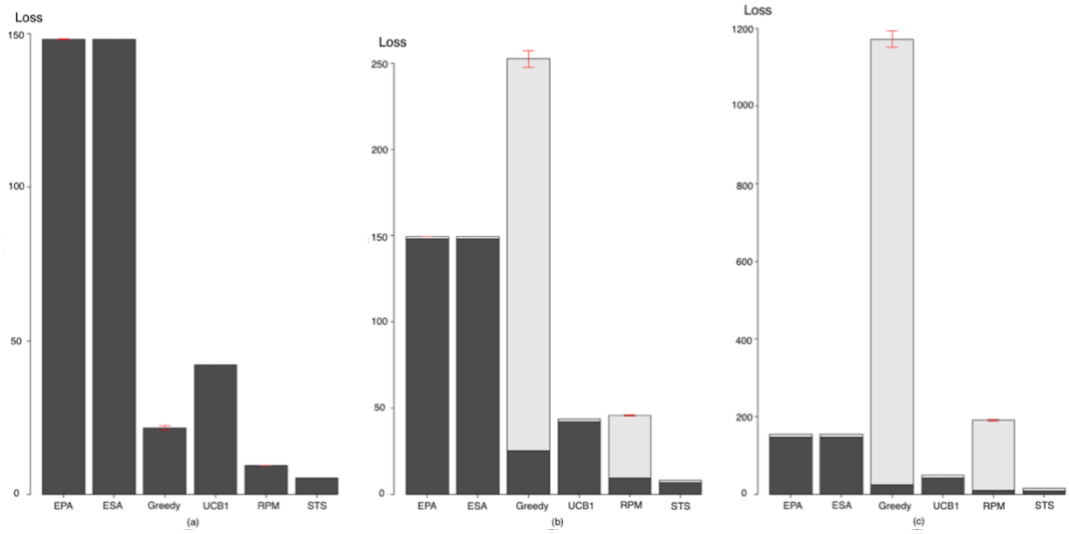


Figure 23. Cumulative regrets (Time Period Length of Experimental Stage = 100, Prior Observation = 5000) under **(a)** Future period = 0 **(b)** Future period = 1000 **(c)** Future period = 5000. Black bar is cumulative regrets during experiments stage and grey bar is cumulative virtual future regrets.

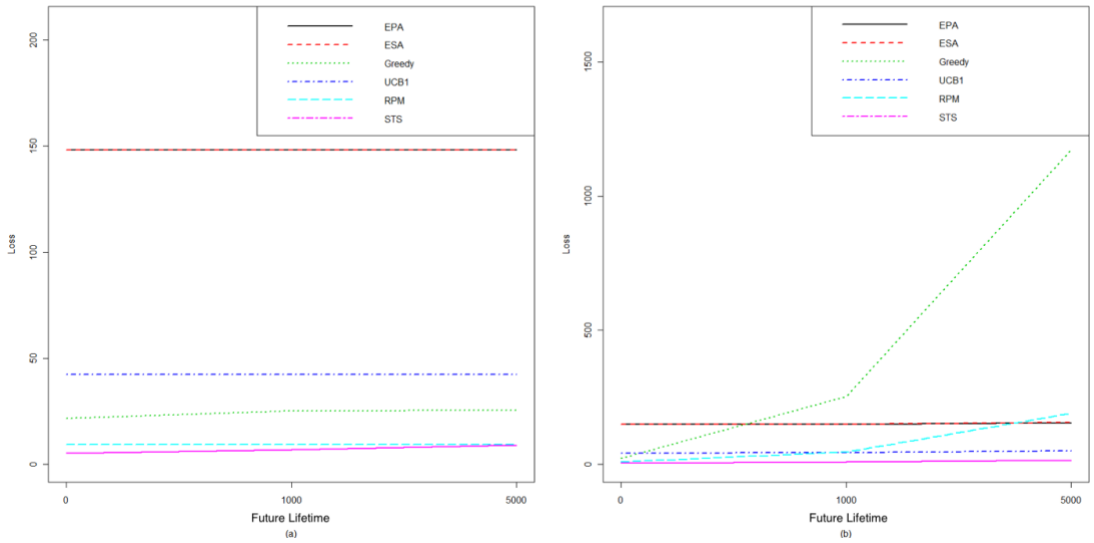


Figure 24. How does Future period change regrets (Time Period Length of Experimental Stage = 100, Prior Information Size = 5000) (a) Cumulative regrets during experiments stage (b) Total cumulative regrets

From the experimental results in Table 5, 6, 7, 8 and Figure 17, 18, 19, 20, 21, 22, 23, 24 we can draw the following conclusions:

- Among all the algorithms, our proposed sequential two stages (STS) algorithm has the smallest mean value of regret, whatever the prior observation size, the values of time period length in experimental stage, and the future period of post-experimental stage are.

- Equal size allocation (ESA) performs the worst if the future period of post-experimental stage is small. As future period gets larger, the performance turns to be better than other algorithms. This is because that ESA focuses on reducing

statistical uncertainty, which is close to be optimal if the future period of post-experimental stage is large.

- On the contrary, pure greedy algorithm performs the worst when the future period of post-experimental stage is large. The larger the future period of post-experimental stage is, the worse it performances. This is because that the pure greedy algorithm suffers from the highest statistical uncertainty for the result, which leads to huge loss when the period of experimental stage is large.

- Without any prior information, equal probability allocation (EPA) is the same as equal size allocation (ESA). With prior information, EPA usually performs worse than ESA. When the period of experimental stage is larger, or the period length of experimental stage is smaller, the difference between EPA and ESA is bigger

- UCB1 and randomized probability matching (RPM) algorithms are neither like ESA, which reduces statistical uncertainty without considering reward, nor like pure greedy algorithm, which only maximizes instant reward. From high to low in terms of greedy level, the algorithms are pure greedy algorithm, RPM, STS, UCB1, and ESA.

- Our proposed sequential two stages (STS) algorithm is the only algorithm that takes account of the periods of post-experimental stage in decision making during experimental stage. From Figure 18, 20, 22, 24, we observe that only STS algorithm adaptively achieves a good trade-off for statistical uncertainty and

accumulating reward within experimental stage. If the future period of post-experimental stage is larger, STS tends to allocate traffic in a more balanced way to reduce statistical uncertainty. Otherwise, it tends to be greedier to obtain more instant reward.

3.4.4 Multiple items multiple stages (m×*k*)

The following results are obtained from the simulation with 40 items. It shows that our proposed methods, especially the method 2, consistently perform better than the baseline methods in term of average reward.

Table 9: Average reward Lift compared with A/B test

$(\alpha_{i,0}, \beta_{i,0},)$=(1, 99), number of batches= 500, batch size = 100

Algorithms	Post-experiment size=20k	Post-experiment size=100k	Post-experiment size=500k	Post-experiment size=2.5m
EPA (A/B testing)	0.00997491	0.01026056	0.01040063	0.01087154
Greedy	0.01000172	0.01011643	0.01016987	0.01024791
RPM	0.01012779	0.01018864	0.01023328	0.01031649
Sequential Two Stages (STS) Method 1	0.01006102	0.01011296	0.01018327	0.01027443
Sequential Two Stages (STS) Method 2	0.01013172	0.01028382	0.01047993	0.01088801

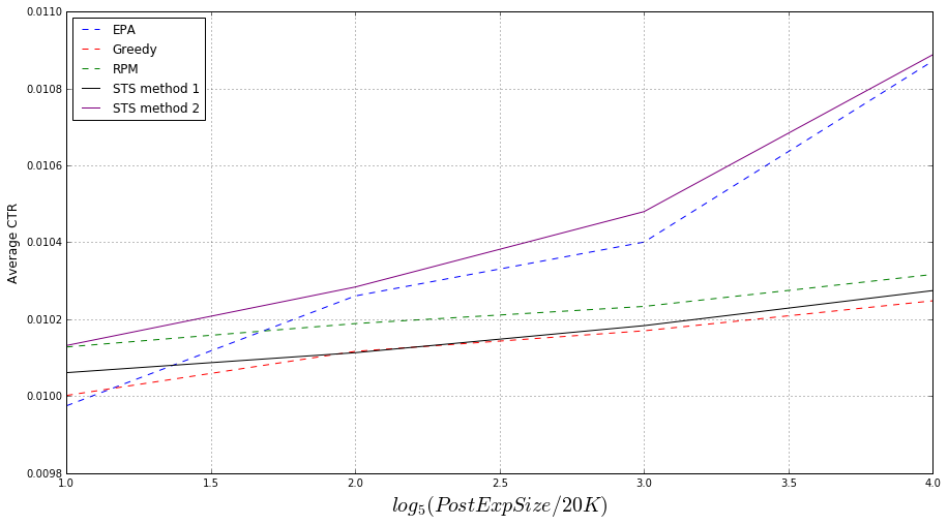


Figure 25. Average reward curve. (α_{10}, β_{10},α_{20}, β_{20})=(1, 99, 1, 99), number of batches = 500, Batch size = 100

CHAPTER IV: Conclusion and Future Work

4.1 Conclusion

In this dissertation, we studied two kinds of the item selection research problems: inexplorable and explorable item selection problems. The optimal policies to solve these two kinds of problems highly depends on the prediction of item values, and prediction uncertainties. We summarize the findings from our research in this chapter. Hopefully, they can inspire new approaches and engage other researchers in better understanding and solving the challenges in this area.

4.1.1 Cascading selection system optimization for inexplorable item selection

To solve the large-scale item selection problem, cascading selection system is widely used in real applications. However, researchers have not satisfactorily addressed the optimization problem for the cascading selection process, e.g., system setting optimization and ranking score optimization. At first glance, these problems seem very simple and straightforward, and do not suggest much research potential in this area. However, they are indeed much more complex and challenging than they may appear.

Our research in this area demonstrates that the performance differs significantly for different system settings, e.g., the number of ranking stages, how many items are filtered at each stage, etc. Through approximate calculations, we provide a final reward estimation equation, which can help us obtain the optimal system settings without tuning them manually. We also provide the solution under infrastructure resource constraints and demonstrate that our solution performs much better than the baseline methods, e.g., one single stage ranking system, through simulation. Beyond that, our solution has already

been used in real-world application and contributes to a huge system performance improvement.

Our initial research demonstrates that there is great potential for incorporating prediction uncertainty into ranking score calculation. There is much work to be done to identify the optimal ranking scoring function to further improve system performance. We will briefly discuss the future work in chapter 4.2.

4.1.2 Total reward optimization for explorable item selection

In this dissertation, we propose an adaptive solution to optimize the total reward in the combined experimental and post-experimental stages. Some existing methods, such as A/B testing, and multi-armed bandit algorithms either ignore the potential loss during the experiment, or only focus on maximizing the reward during the experiment without considering the fact that the result reliability is important for the post-experimental stage usage. Actually, the goal of an experiment is to maximize the total reward, across two stages: experimental stage and post-experimental stage. It is only worth doing significant exploration to obtain a reliable conclusion if the size of post-experimental stage is large enough.

Our proposed adaptive solution to optimize the total reward for experimental stage and post-experimental stage together starts from a simple case: “two items & two-time intervals”. Then, we extend the method to “two items & multiple intervals” and “multiple items & multiple intervals”. The experimental results demonstrate that our proposed algorithm is much more robust than the existing methods, from cases with small post-

experimental size to large post-experimental size. It consistently performs much better than the other compared algorithms under all the tested settings in our simulation.

4.2 Future work

There are many interesting research problems that are worth being further studied for item selection. Here, we list some of them related to our study in this dissertation.

4.2.1 Cascading selection system optimization

4.2.1.1 Cascading system setting optimization

Our approximation provides a reliable estimate for the final reward given system setting. In the approximation, we assume the value of the output of each stage follows a Gaussian distribution. This assumption is accurate when the number of stages is small. As the number of stages increases, the accuracy of this approximation decreases. This is the reason why we observe that our estimation for two-stage problem obtains higher accuracy than three-stage and four-stage problems. If the number of stages is larger than five, we need to find a more accurate estimation method to optimize the system performance.

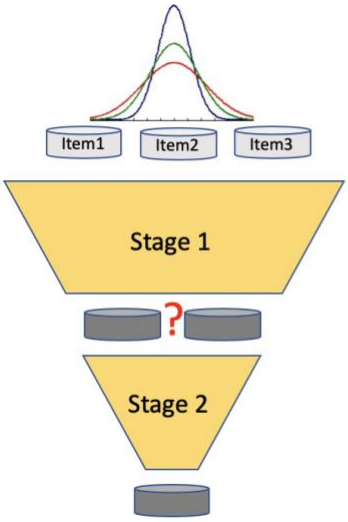
4.2.1.2 Cascading system ranking score optimization

The ranking score optimization problem is actually very complex. Our research is only an initial study in this area. We have discussed how to incorporate prediction uncertainty in calculating the item values for binary event item selection problem. Actually, there are many other ways in which prediction uncertainty can affect cascading selection system efficiency. We illuminate this through the following simple example.

Suppose we have three ads, and the expected return for each of them is $3. Assume that the prediction errors are white Gaussian noise, and the prediction variance is 1, 2 and 5, respectively.

- Case 1: The problem is to select one ad from these three ads. And there is only one single ranking stage. It is obvious that the expected return would be the same no matter which ad we choose.

- Case 2: The problem is still to select one ad from these three ads. But we have two stages. The first stage is to select two from three ads. And the second stage is to select one from these two ads. Suppose the prediction variances from that predictor are 0.8, 1.8, 4.8, respectively. As Figure 26 shows, the best policy would be ad 2 and ad 3. The ads with high uncertainty lead to larger expected



Two selected ads	Expected return
Item1, Item2	$3.15
Item2, Item3	$4.16
Item1, Item3	$3.47

Figure 26. Selecting three items through two stage cascading selection system

To make it more general, we briefly talk about the expected return for a two-item selection problem. Suppose we have two items, the true values of which, y_1 and y_2, are both unknown. The predicted values of these two items are z_1 and z_1. Assume the error from the predictions are Gaussian distributed $N(0, \sigma_1^2)$ and $N(0, \sigma_2^2)$. If we select one item from these two items based on another predictor, the errors of which are also Gaussian distributions $N(0, \sigma'^2_1)$ and $N(0, \sigma'^2_2)$, the expected reward would be:

$$E(r|z_1, z_2, \sigma_1^2, \sigma_2^2, \sigma'^2_1, \sigma'^2_2) = \\ = \mu_1 \cdot \Phi\left(\frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2+\sigma_2^2+\sigma_1'^2+\sigma_2'^2}}\right) + \mu_2 \cdot \left(1 - \Phi\left(\frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2+\sigma_2^2+\sigma_1'^2+\sigma_2'^2}}\right)\right) \\ + \frac{\sigma_1^2+\sigma_2^2}{\sqrt{\sigma_1^2+\sigma_2^2+\sigma_1'^2+\sigma_2'^2}} \phi\left(\frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2+\sigma_2^2+\sigma_1'^2+\sigma_2'^2}}\right) \quad (50)$$

The calculation would be much more complex if there are more items, which needs to be further studied.

4.2.1.3 Joint optimization of system settings and ranking score

As mentioned in above section, we contribute to discovering that the optimal ranking score for cascading selection problem is not as simple as researchers and engineers thought before. However, deeper research still requires further efforts. After obtaining a clear and deep understanding of ranking score optimization, it would be worth studying the joint optimization of system settings and ranking score, as these two optimization problems are closely related to each other. For example, the optimal system settings would change if we use a different ranking score function to rank and select items. To get a true optimal solution, we should jointly optimize the system for these two problems.

4.2.2 Online experiment optimization

4.2.2.1 Contextual information

In our study, each sample is assumed to be the same. If these samples have some contextual features, this assumption would not hold. In real applications, cookies could be used to store users' information, including browser type, visiting time, etc. How to use this information to improve the efficiency of online experiment is also an interesting topic for research.

4.2.2.2 Nondeterministic future traffic

In real-world applications, experimental traffic is usually unknown, which needs to be estimated through time series methods. This situation has not been addressed in this dissertation. The impact of the uncertainty of future traffic estimation also needs to be furthered studied.

4.2.2.3 Item budget

We have not accounted for the budget of each item in our study. In real-world applications, e.g., display of ads, each campaign has its own budget. How to efficiently make the trade-off for exploration and exploitation given campaign budget is also an interesting research topic. Intuitively, if a campaign has a larger budget, we should be able to explore more. And meanwhile, it is also worth exploring more for these larger budget campaigns, as they have larger delivery demand in the post-experimental stage.

APPENDICES

Appendix A

The predictions from the first stage follows a Gaussian distribution $\hat{y}_{i,1} \sim N(\mu, \sigma^2 + \sigma_1^2)$. Now, let us show how to obtain the distribution of the *i*th order statistic of the predictions.

For the value of the *i*th order statistic of a random variable $\hat{y}_{i,1}$ to be equal to x, three conditions need to be meet:

- Condition 1: i-1 values of $\hat{y}_{i,1}$ need to be less than x, of which the probability is $\Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)$.
- Condition 2: n-i values of $\hat{y}_{i,1}$ need to be larger than x, of which the probability is $1-\Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)$.
- Condition 3: One value of $\hat{y}_{i,1}$ needs to be equal to x, of which the probability is $\phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)$,

As there are $n \cdot \binom{n-1}{i-1}$ options to make the choice, the probability of the value of the *i*th order statistic of a random variable $\hat{y}_{i,1}$ to be equal to x is

$$P_{\hat{y}_{i,1}}(x|\mu, \sigma^2, \sigma_1^2, n) = \frac{n!}{(n-i)!(i-1)!} \Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)^{n-i} \cdot \left(1-\Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)\right)^{i-1} \cdot \phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right). \quad (51)$$

If we take the mean value of $\hat{y}_{i,1}$, we have

$$E(\hat{y}_{i,1}|\mu,\sigma^2,\sigma_1^2,n)=\int_{-\infty}^{\infty}x\cdot\frac{n!}{(n-i)!(i-1)!}\Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)^{n-i}\cdot\left(1-\Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)\right)^{i-1}\cdot\phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)\cdot dx. \quad (52)$$

Let us denote $p_i=\Phi\left(\frac{x-\mu}{\sqrt{\sigma^2+\sigma_1^2}}\right)$. Then the expectation of $\hat{y}_{i,1}$ is

$$\begin{aligned}E(\hat{y}_{i,1}|\mu,\sigma^2,\sigma_1^2,n)&=\int_{-\infty}^{\infty}x\cdot\frac{n!}{(n-i)!\,(i-1)!}\cdot {p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot\frac{dp_i}{dx}\cdot dx\\&=\int_0^1 x\cdot\frac{n!}{(n-i)!\,(i-1)!}\cdot {p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot dp_i\\&=\int_0^1\left(\sqrt{\sigma^2+\sigma_1^2}\cdot\Phi^{-1}(p_i)+\mu\right)\cdot\frac{n!}{(n-i)!\,(i-1)!}{p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot dp_i\\&=\mu+\sqrt{\sigma^2+\sigma_1^2}\cdot\int_0^1\Phi^{-1}(p_i)\cdot\frac{n!}{(n-i)!\,(i-1)!}\cdot {p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot dp_i\\&=\mu+\sqrt{\sigma^2+\sigma_1^2}\cdot E_{Beta}(p_i|i,n-i+1)\Phi^{-1}(p_i)\\&=\mu+\sqrt{\sigma^2+\sigma_1^2}\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right).\end{aligned} \quad (53)$$

Similarly, we have

$$\begin{aligned}E({\hat{y}_{i,1}}^2|\mu,\sigma^2,\sigma_1^2,n)&=\int_{-\infty}^{\infty}x^2\cdot\frac{n!}{(n-i)!\,(i-1)!}\cdot {p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot\frac{dp_i}{dx}\cdot dx\\&=\int_{-\infty}^{\infty}x^2\cdot\frac{n!}{(n-i)!\,(i-1)!}\cdot {p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot dp_i\\&=\int_{-\infty}^{\infty}\left(\sqrt{\sigma^2+\sigma_1^2}\cdot\Phi^{-1}(p_i)+\mu\right)^2\cdot\frac{n!}{(n-i)!\,(i-1)!}{p_i}^{n-i}\cdot(1-p_i)^{i-1}\cdot dp_i\\&=\mu^2+2\sqrt{\sigma^2+\sigma_1^2}\cdot\mu\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)+(\sigma^2+\sigma_1^2)\cdot E_{Beta(p_i|i,n-i+1)}\left(\Phi^{-1}(p_i)\right)^2\\&=\mu^2+2\sqrt{\sigma^2+\sigma_1^2}\cdot\mu\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)+(\sigma^2+\sigma_1^2)\\&\cdot\left(\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)^2+\frac{i\cdot(n-i+1)}{(\sigma^2+\sigma_1^2)\cdot(n+1)^2\cdot(n+2)\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}\right)\end{aligned} \quad (54)$$

Then the variance of $\hat{y}_{i,1}$ is

$$\begin{aligned}Var(\hat{y}_{i,1}|\mu,\sigma^2,\sigma_1^2,n)&=E({\hat{y}_{i,1}}^2|\mu,\sigma^2,\sigma_1^2,n)-\left(E(\hat{y}_{i,1}|\mu,\sigma^2,\sigma_1^2,n)\right)^2\\&=\frac{i\cdot(n-i+1)}{(n+1)^2\cdot(n+2)\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}.\end{aligned} \quad (55)$$

So approximately, the distribution of $\hat{y}_{i,1}$ is a Gaussian distribution

$$N\left(\mu+\sqrt{\sigma^2+\sigma_1^2}\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right),\frac{i\cdot(n-i+1)}{(n+1)^2\cdot(n+2)\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}\right). \tag{56}$$

Appendix B

As we know, the true value of the first stage output (also the second stage input) y_1 is an equal weighted mixture of a series of Gaussian distributions:

$$N\left(\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}},\frac{\sigma^2\cdot\sigma_1^2}{\sigma^2+\sigma_1^2}+\frac{i\cdot(n-i+1)}{(n+1)^2\cdot(n+2)\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}\right), \tag{57}$$

where $i=1,2,\ldots,k_1$.

As it is an equal weighted mixture, the probability density function (PDF) of the distribution is

$$p(y_1|\mu,\sigma^2,\sigma_1^2,n)=\frac{1}{k_1}\cdot\sum_{i=1}^{k_1}\phi\left(\frac{x-\left(\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}\right)}{\sqrt{\frac{\sigma^2\cdot\sigma_1^2}{\sigma^2+\sigma_1^2}+\frac{i\cdot(n-i+1)}{(n+1)^2\cdot(n+2)\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}}}\right). \tag{58}$$

Then, the expectation of y_1 is

$$\begin{aligned}E(y_1|\mu,\sigma^2,\sigma_1^2,n)&=\int_{-\infty}^{\infty}x\cdot\frac{1}{k_1}\cdot\sum_{i=1}^{k_1}\phi\left(\frac{x-\left(\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}\right)}{\sqrt{\frac{\sigma^2\cdot\sigma_1^2}{\sigma^2+\sigma_1^2}+\frac{i\cdot(n-i+1)}{(n+1)^2\cdot(n+2)\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2}}}\right)\cdot dx\\&=\frac{1}{k_1}\cdot\sum_{i=1}^{k_1}E\left(y_{i,1}|\mu,\sigma^2,\sigma_1^2,n\right)=\frac{1}{k_1}\cdot\sum_{i=1}^{k_1}\left(\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{k_1\sqrt{\sigma^2+\sigma_1^2}}\right)\\&=\mu+\frac{\sigma^2\cdot\sum_{i=1}^{k_1}\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{k_1\sqrt{\sigma^2+\sigma_1^2}}.\end{aligned} \tag{59}$$

The variance of y_1 is

$$
\begin{aligned}
Var(y_1|\mu,\sigma^2,\sigma_1^2,n) &= E({y_1}^2|\mu,\sigma^2,\sigma_1^2,n) - \left(E(y_1|\mu,\sigma^2,\sigma_1^2,n)\right)^2 \\
&= \frac{1}{k_1}\cdot\sum_{i=1}^{k_1} Var\left(y_{i,1}|\mu,\sigma^2,\sigma_1^2,n\right) \\
&+ \sum_{i=1}^{k_1}\left(E\left(y_{i,1}|\mu,\sigma^2,\sigma_1^2,n\right) - E(y_1|\mu,\sigma^2,\sigma_1^2,n)\right)^2 \\
&= \frac{\sigma^2\cdot\sigma_1^2}{\sigma^2+\sigma_1^2} + \sum_{i=1}^{k_1}\frac{i\cdot(n-i+1)}{(n+1)^2\cdot(n+2)\cdot k_1\cdot\left(\phi\left(\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)\right)\right)^2} \\
&+ \sum_{i=1}^{k_1}\left(\mu+\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}\right)^2 - \left(\frac{\sigma^2\cdot\sum_{i=1}^{k_1}\Phi^{-1}\left(\frac{n-i+1}{n+1}\right)}{k_1\sqrt{\sigma^2+\sigma_1^2}}\right)^2.
\end{aligned}
\tag{60}
$$

Appendix C

In the dissertation, we assume the values of items are independent with each other, which is not always true in real-world applications. We would like to discuss the impact of dependency in our cascading selection problem. To be specific, we want to measure the accuracy of equation (14) in Chapter 2. Now, let us verify how the accuracy of our calculation in equation (14) is impacted by item correlations.

$$
V_1(S_k) = k_1\cdot\mu + \sum_{i=1}^{k_1\cdot}\frac{\sigma^2\cdot\Phi^{-1}\left(\frac{n-i-\alpha+1}{n-2\alpha+1}\right)}{\sqrt{\sigma^2+\sigma_1^2}}. \tag{61}
$$

General Case

Assuming the covariance matrix of item values is S, i.e.

$$
S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{bmatrix}. \tag{62}
$$

It can also be represented in correlation parameters form, i.e.

$$
S = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \sigma_n \end{bmatrix}\cdot\begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \cdots & \rho_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{n1} & \cdots & \cdots & 1 \end{bmatrix}\cdot\begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \sigma_n \end{bmatrix}. \tag{63}
$$

Here, $\rho_{ij} = \rho_{ji}$, which is the correlation coefficients, which is the correlation coefficients between item *i* and *j*. It is obvious that

$$s_{ii} = \sigma_i^2, \tag{64}$$

$$s_{ij} = \rho_{ij} \cdot \sigma_i \cdot \sigma_j. \tag{65}$$

We define the mean absolute correlation coefficient (MACC) as

$$\rho_{MACC} = \frac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \rho_{ij}. \tag{66}$$

A larger ρ_{MACC} indicates more correlation between item values. Figure 27 shows the absolute error of the estimation in equation (14) given different ρ_{MACC}.

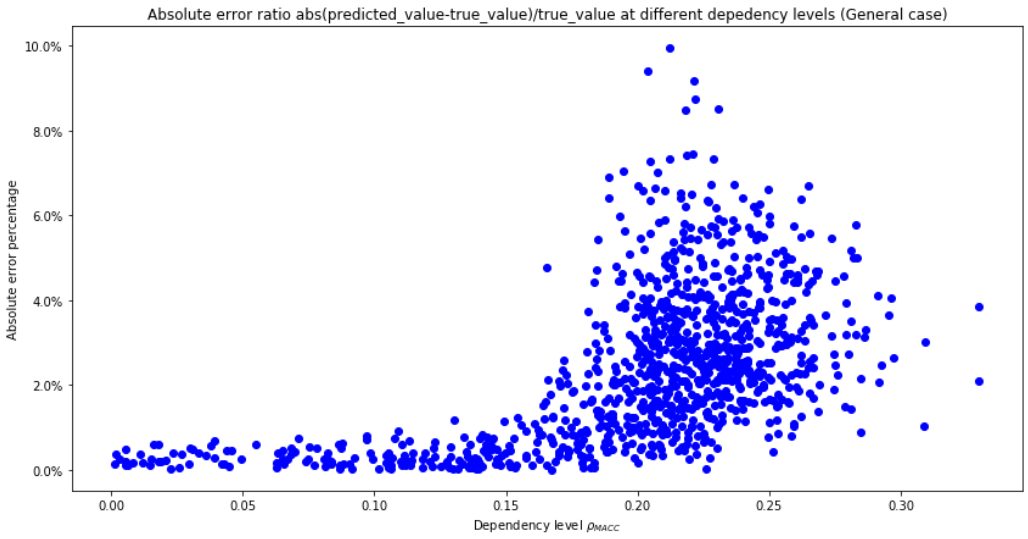


Figure 27. Absolute error given different correlation levels

It shows that when $\rho_{MACC} < 0.15$, the prediction from equation (14) is quite accurate, with error less than 1%. The error increases significantly when $\rho_{MACC} > 0.15$. To verify whether equation (14) is good enough to be used in real-world applications, we collect some data points from real advertising problem with millions of samples and observe that

the typical ρ_{MACC} is around 0.06, which means the value calculation in equation (14) is accurate in these advertising problems.

We also check the situation that the correlations between these item values are unbalanced distributed. For example, if 30% of items are independent with other items, the absolute error ratio is still very low when ρ_{MACC} is less than 0.10, which is shown in Figure 28.

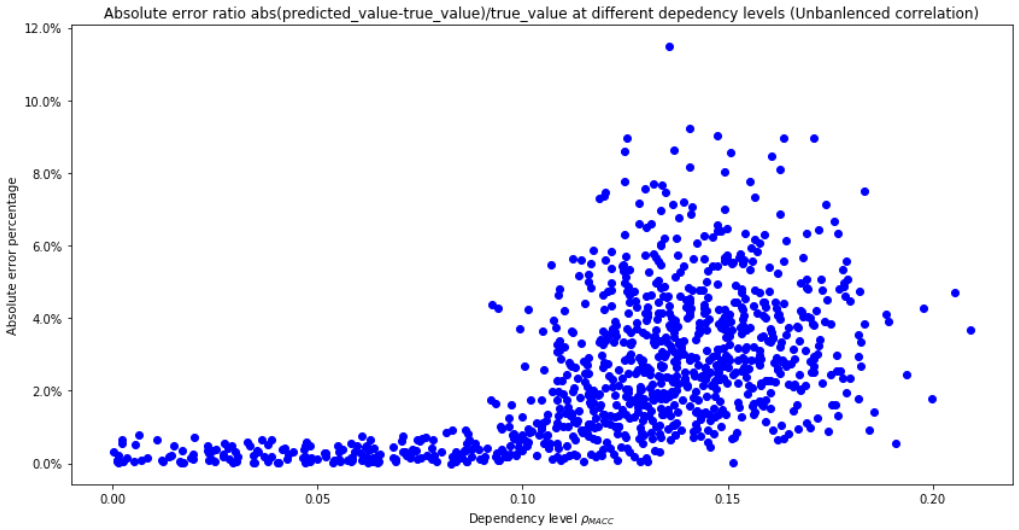


Figure 28. Absolute error given different correlation levels in unbalanced case (30% items are independent with others)

Hierarchical Bayesian model

In real-world applications, one common scenario is that the items within the same category are highly correlated, and two items from different categories are much less correlated. Taking advertising for example, for a user, the click through rates (CTRs) of shoes, no matter Nike, Adidas, Puma, are close to each other, CTRs of Best Buys, Fry's electronics are also close. The correlations between these categories, e.g. shoes, electronic

store, much smaller than the correlations within each category. If we look at this problem from a generative way, it is more like a two-step process as follows in Figure 29.

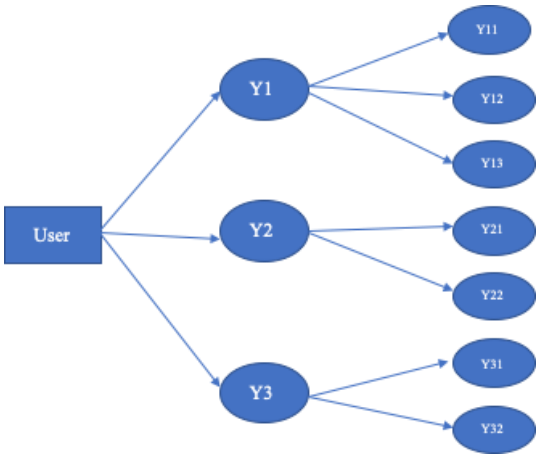


Figure 29. Two-stage generative process of users' CTRs

Here, y_1, y_2, y_3 are users' click through rates (CTR) for ads categories, i.e. cars and shoes. y_{11} is the CTR of BMW ads, y_{12} is the CTR of Audi ads, and y_{13} is the CTR of Toyota ads. y_{21} is the CTR of Nike shoes, y_{21} is the CTR of Puma shoes. There are strong correlations within categories, e.g. y_{11} and y_{12}, and weak correlations between categories, e.g. y_{11} and y_{21}. Intuitively, if a user has a high CTR of Nike shoes, he/she is also likely to click Puma shoes. Meanwhile, the CTR of a user clicking shoes ads is almost independent to the CTR of a user clicking automobile ads.

Now, let us redefine a subset selection problem firstly. Suppose we have n_c categories with the true value of y_i with i.i.d. Gaussian distribution, i.e., $y_i \sim N(\mu, \sigma'^2)$, $i = 1, 2, \dots, n_c$. For each category, there are m_c items with true value of $y_{i,j}$, $j = 1, 2, \dots, m_c$. So, the number of items is $n_c \cdot m_c$, which is equal to n in the dissertation. Suppose $y_{i,j} = y_i + \Delta_{i,j}$, where $\Delta_{i,j}$ is Gaussian distributed as $\Delta_{i,j} \sim N(0, \sigma_\Delta^2)$. Here, our assumption is a simplification of real-world problem, where y_i should be weakly correlated with each other.

Different categories can also have different number of items. The graphical representation of above process is as below in Figure 30.

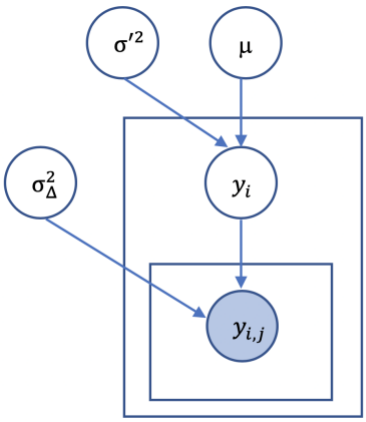


Figure 30. Hierarchical Bayesian graphical model

Now, let us check how the accuracy of our calculation in equation (14) is impacted by the within category correlations. Note that σ^2 in equation (14) is calculated as the variance of $y_{i,j}$.

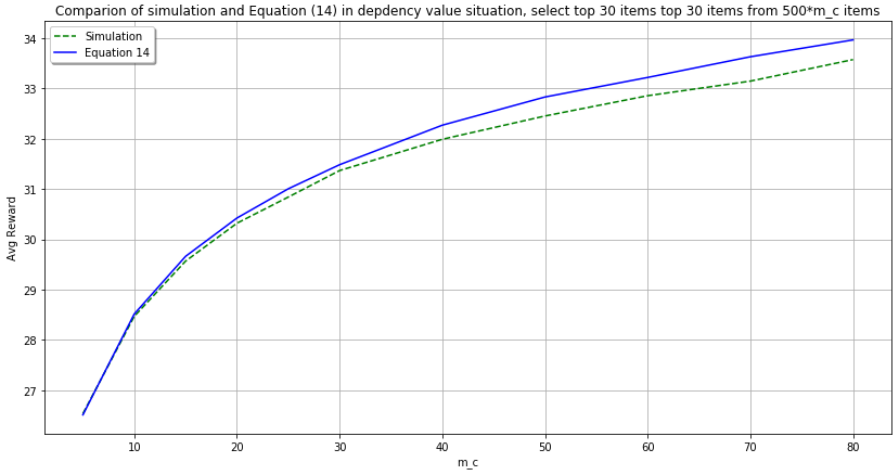


Figure 31. The plot of performance VS m_c, μ = 0, $\sigma'^2 = 64$, $\sigma_\Delta^2 = 16$, $\sigma_1^2 = 1$, $n = 1000$, $n_c = 200$

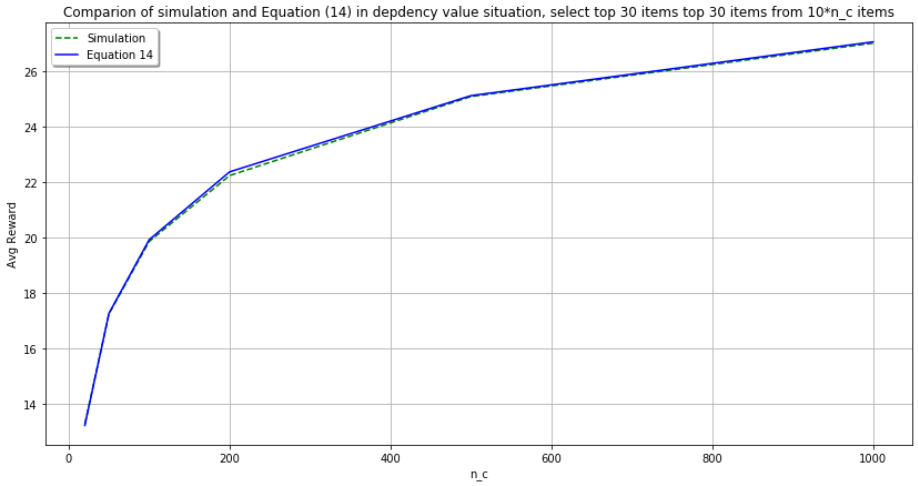


Figure 32. The plot of performance VS n_c, μ = 0, $\sigma'^2 = 64$, $\sigma_\Delta^2 = 16$, $\sigma_1^2 = 1$, $n = 1000$,$m_c = 10$

The simulation results are shown in Figure 31 and Figure 32. It is observed that the accuracy of equation (14) is sensitive to the value of m_c. The accuracy is very bad when m_c is large. Fortunately, in our real advertising applications, the typical value of m_c for a user is usually less than 20 because of ads targeting requirement. So, our equation (14) still works well in the "dependent items" situation.

We also plot the absolute error ratio in hierarchical Bayesian process case in Figure 33, which shows a similar conclusion that the error is very small when ρ_{MACC} is less than 0.10.

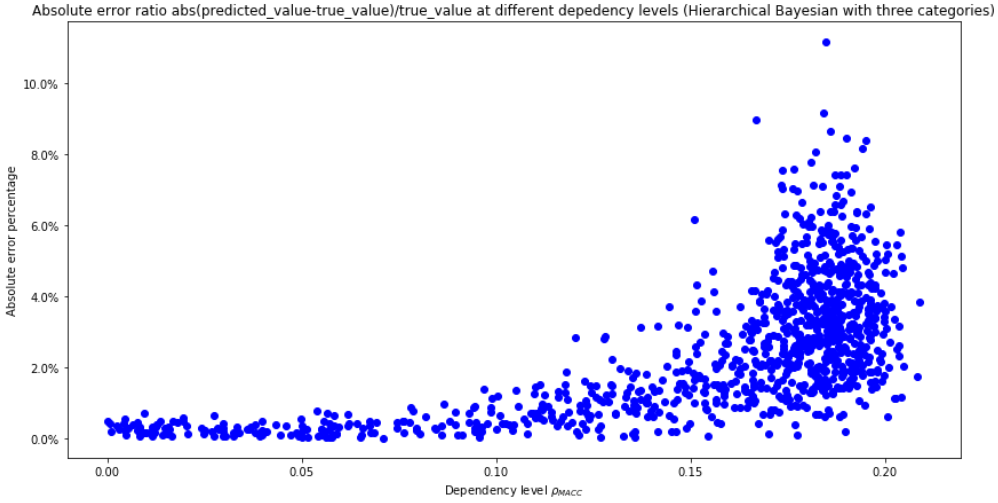


Figure 33. Absolute error given different correlation levels in hierarchical Bayesian case (Three categories)

So, we conclude that the calculation in equation (14) is accurate enough to be used in most of real-world applications, even when the item values are not truly independent with each other.

BIBLIOGRAPHY

[1] Number of apps available in leading app stores as of 3rd quarter 2018 https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[2] These charts show how far Google and Facebook are ahead of Twitter. https://www.businessinsider.com/macquarie-research-facebook-google-and-twitter-number-of-advertisers-2015-2

[3] Luce, R. D. (1959). Individual choice behavior: A theoretical analysis. Wiley

[4] Gupta, S.S. and Miescke, K.J. Sequential selection procedures – A decision-theoretic approach, Ann. Statist. 12: 336-350

[5] M.A. Fligner and J.S. Verducci. Multistage ranking models. Journal of the American Statistical Association, 83:892–901, 1988.

[6] Robert E. Bechhofer, Thomas J. Santner, and David Goldsman. Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons. Wiley, New York, 1995.

[7] Batur, D., and S.-H. Kim. 2006. Fully sequential selection procedures with parabolic boundary. IIE Transactions 38:749–764.

[8] Kim, S.-H., B. L. Nelson. 2007. Recent advances in ranking and selection. Proceedings of the 2007 Winter Simulation Conference, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey. 162-172.

[9] Expected Normal Order Statistics (Exact and Approximate) J. P. Royston. Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 31, No. 2 (1982), pp. 161-165

[10] Wikipedia, Design of experiments.

http://en.wikipedia.org/wiki/Experimental_design

[11] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. Data Mining and Knowledge Discovery, 18(1), 2009.

[12] R. Kohavi, R M. Henne, D. Sommerfield, Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO. Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD2007), pp. 959-967.

[13] R. Kohavi, Ron, Crook, Thomas and Longbotham, Roger. Online Experimentation at Microsoft. Third Workshop on Data Mining Case Studies and Practice Prize. 2009.

[14] R. Kohavi, Ron, et al. Controlled experiments on the web: survey and practical guide, Data Mining and Knowledge Discovery. February 2009, Vol. 18, 1, pp. 140-181.

[15] Chatham B, Temkin BD, Amato M (2004) A primer on A/B testing. Forrester Research

[16] Eisenberg B, Eisenberg J (2005) Call to action, secret formulas to improve online results. Wizard Academy Press, Austin, 2005. Making the dial move by testing, introducing A/B testing

[17] Nielsen J (2005) Putting A/B testing in its place. Useit.com Alertbox. Aug 15, 2005.

http://www.useit.com/alertbox/20050815.html

[18] Kyle Rush, Optimization at the Obama campaign: a/b testing. http://kylerush.net/blog/optimization-at-the-obama-campaign-ab-testing/

[19] Steven L. Scott, Multi-armed bandit experiments. https://support.google.com/analytics/answer/2844870?hl=en

[20] Chris Stucchio, Why Multi-armed Bandit algorithms are superior to A/B testing

http://www.chrisstucchio.com/blog/2012/bandit_algorithms_vs_ab.html

[21] T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. Advances in applied mathematics, 6(1):4{22, 1985.

[22] Watkins, C.: Learning from Delayed Rewards. PhD thesis, (1989) University of Cambridge, Cambridge, England

[23] Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: Proceedings of the 16th European Conference on Machine Learning (ECML'05), Porto, Portugal (2005) 437-448

[24] S. Mannor and J. N. Tsitsiklis. The Sample Complexity of Exploration in the Multi-armed Bandit Problem. In Sixteenth Annual Conference on Computational Learning Theory (COLT), 2003.

[25] Nicolo Cesa-Bianchi and Paul Fischer. Finite-time regret bounds for the multi-armed bandit problem. In Proceedings of ICML, pages 100-108, 1998.

[26] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite Time Analysis of the Multi-armed Bandit Problem. Machine Learning, 47(2/3):235-256, 2002.

[27] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems, October 2010. http://www.cs.mcgill.ca/~vkules/bandits.pdf.

[28] D. Luce. Individual Choice Behavior. Wiley, 1959.

[29] Richard S. Sutton and Andrew G. Barto. Reinfocement Learning: An Introduction. MIT Press, 1998.

[30] L. Kocsis and Csaba Szepesvari. Bandit based monte-carlo planning. In Proceedings of ECML, pages 282-293, 2006.

[31] Jean-Yves Audibert, Remi Munos, and Csaba Szepesvari. Exploration-exploitation trade-off using variance estimates in multi-armed bandits. Theoretical Computer Science, 2009.

[32] Agrawal, R. (1995). Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. Advances in Applied Probability, 27, 1054–1078.

[33] Robert D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In Advances in Neural Information Processing Systems 17, pages 697–704. MIT Press, 2005.

[34] P. Auer and R. Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. Periodica Mathematica Hungarica, 61(1-2):55–65, 2010.

[35] A. Garivier and O. Cappe. The KL-UCB algorithm for bounded stochastic bandits and beyond. In Proceedings of COLT, 2011.

[36] Gittins, J. C. (1979): Bandit Processes and Dynamic Allocation Indices. Journal of the Royal Statistical Society, Volume 41(2), pages 148-177.

[37] Gittins, J. C. (1989): Multi-armed Bandits Allocation Indices. Wiley, New York

[38] Varaiya, P., Walrand, J. and Buyukkoc, C. (1985): Extensions of the multi-armed bandit problem: The discounted case. IEEE Transactions on Automatic Control, Volume AC-30, pages 426-439

[39] Richard R. Weber (1992). On the Gittins Index for Multi-armed Bandits. The Annals of Applied Probability 2 (4): 1024–1033.

[40] Powell WB. Approximate Dynamic Programming: Solving the Curses of Dimensionality. Wiley: New York, 2007.

[41] P. Whittle, Multi-armed bandits and the Gittins index, Journal of the Royal Statistical Society, Series B (Methodological), vol. 42, no. 2, pp. 143-149, 1980.

[42] M. Brezzi and T. L. Lai, Optimal learning and experimentation in bandit problems, Journal of Economic Dynamics and Control, vol. 27, no. 1, pp. 87-108, 2002.

[43] Brezzi M, Lai TL. Incomplete learning from endogenous data in dynamic allocation. Econometrica 2000; 68(6):1511–1516.

[44] L. Dai, R. Akella, Lifetime reward optimization for online ads experiment, International conference on machine learning and data mining, July 20-25, 2019, New York, USA

[45] L. Dai, R. Akella, Waterfall ranking and filtering optimization, ISSAT International Conference on Data Science and Intelligent Systems (DSIS), Aug 1-3, 2019, Las Vegas, Nevada, USA

[46] L. Dai, R. Online Controlled Experiment Design: Trade-off between Statistical Uncertainty and Cumulative Reward, Master Thesis, Jun 2014, University of California Santa Cruz, Santa Cruz, California, USA

[47] T. J. Sullivan. Introduction to Uncertainty Quantification. Texts in Applied Mathematics. Springer International Publishing, 2015.